

BEST AVAILABLE COPY

1500314

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE

United States Patent and Trademark Office

March 23, 2005

THIS IS TO CERTIFY THAT ANNEXED HERETO IS A TRUE COPY FROM THE RECORDS OF THE UNITED STATES PATENT AND TRADEMARK OFFICE OF THOSE PAPERS OF THE BELOW IDENTIFIED PATENT APPLICATION THAT MET THE REQUIREMENTS TO BE GRANTED A FILING DATE.

APPLICATION NUMBER: 60/499,961

FILING DATE: *September 02, 2003*

RELATED PCT APPLICATION NUMBER: *PCT/US04/28926*



Certified by

Under Secretary of Commerce
for Intellectual Property
and Director of the United States
Patent and Trademark Office

17707 U.S. PTO
09/02/03

PTO/SB/16 (08-03)

Approved for use through 07/31/2006. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PROVISIONAL APPLICATION FOR PATENT COVER SHEET

This is a request for filing a PROVISIONAL APPLICATION FOR PATENT under 37 CFR 1.53(c).

Express Mail Label No. ER447520883US

22389 U.S. PTO
60/499961

INVENTOR(S)					
Given Name (first and middle (if any))		Family Name or Surname		Residence (City and either State or Foreign Country)	
PAUL HENRY		UNDERBRINK FALK		Lake Forest, CA Long Beach, CA	
Additional inventors are being named on the _____ 1 _____ separately numbered sheets attached hereto					
TITLE OF THE INVENTION (500 characters max)					
A GPS SYSTEM					
Direct all correspondence to: CORRESPONDENCE ADDRESS					
<input type="checkbox"/> Customer Number: 					
OR					
<input checked="" type="checkbox"/> Firm or Individual Name		Francisco A. Rubio-Campos			
Address		The Eclipse Group			
Address		26895 Aliso Creek Road, Suite B-104			
City		Aliso Viejo	State	CA	Zip 92656
Country		USA	Telephone	(949) 448-9410	Fax (714) 948-8903
ENCLOSED APPLICATION PARTS (check all that apply)					
<input checked="" type="checkbox"/> Specification Number of Pages 202		<input type="checkbox"/> CD(s), Number _____			
<input checked="" type="checkbox"/> Drawing(s) Number of Sheets 117		<input type="checkbox"/> Other (specify) _____			
<input type="checkbox"/> Application Date Sheet. See 37 CFR 1.76					
METHOD OF PAYMENT OF FILING FEES FOR THIS PROVISIONAL APPLICATION FOR PATENT					
<input checked="" type="checkbox"/> Applicant claims small entity status. See 37 CFR 1.27. <input checked="" type="checkbox"/> A check or money order is enclosed to cover the filing fees. <input checked="" type="checkbox"/> The Director is hereby authorized to charge filing fees or credit any overpayment to Deposit Account Number: 502542 <input type="checkbox"/> Payment by credit card. Form PTO-2038 is attached.					FILING FEE Amount (\$) <div style="border: 1px solid black; padding: 10px; width: 100px; margin: 0 auto;">80.00</div>
The invention was made by an agency of the United States Government or under a contract with an agency of the United States Government. <input checked="" type="checkbox"/> No. <input type="checkbox"/> Yes, the name of the U.S. Government agency and the Government contract number are: _____					

[Page 1 of 2]

Respectfully submitted,

SIGNATURE

TYPED or PRINTED NAME Francisco A. Rubio-Campos

TELEPHONE (949) 448-9410

Date September 2, 2003

REGISTRATION NO. 45,358

(if appropriate)

Docket Number: ST02042USV

USE ONLY FOR FILING A PROVISIONAL APPLICATION FOR PATENT

This collection of information is required by 37 CFR 1.51. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 8 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Mail Stop Provisional Application, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

PROVISIONAL APPLICATION COVER SHEET
Additi nal Pag

PTO/SB/16 (08-03)

Approved for use through 07/31/2006. OMB 0651-0032

U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

Docket Number ST02042USV

INVENTOR(S)/APPLICANT(S)		
Given Name (first and middle (if any))	Family or Surname	Residence (City and either State or Foreign Country)
MANGESH	CHANSARKAR	Sunnyvale, CA
SUNDAR	RAMAN	San Jose, CA
CHARLES	NORMAN	Huntington Beach, CA
STEVEN	GRONEMEYER	Cedar Rapids, IA
ROBERT	TSO	South San Gabriel, CA
NICOLAS	VANTALON	San Jose, CA
CHITTHARANJAN	DASANNACHARYA	Irvine, CA
VOYA	PROTIC	San Jose, CA

[Page 2 of 2]

Number 2 of 2

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

A GPS SYSTEM

INVENTORS

PAUL UNDERBRINK

HENRY FALK

5 MANGESH CHANSARKAR

SUNDAR RAMAN

CHARLES NORMAN

STEVEN GRONEMEYER

ROBERT TSO

10 NICOLAS VANTALON

CHITTHARANJAN DASANNACHARYA

VOYA PROTIC

15 **BACKGROUND OF THE INVENTION**

[001] 1. Field of the Invention.

[002] This invention relates generally to positioning systems. More specifically, this invention relates to methods and systems for implementing signal processing control and feature subsystems for processing global positioning system signals.

20

[003] 2. Related Art.

[004] The worldwide utilization of wireless devices such as two-way radios, pagers, portable televisions, personal communication system ("PCS"), personal digital assistants ("PDAs") cellular telephones (also known a "mobile phones"), Bluetooth, satellite radio

receivers and Satellite Positioning Systems ("SPS") such as the Global Positioning Systems ("GPS"), also known as NAVSTAR, is growing at a rapid pace. Current trends are calling for the incorporation of SPS services into a broad range of electronic devices and systems, including Personal Digital Assistants (PDAs), cellular telephones, portable computers,
5 automobiles, and the like. Manufacturers constantly strive to reduce costs and produce the most cost-attractive product possible for consumers.

[005] At the same time, the manufacturers attempt to provide a product as rich in features, and as robust and reliable, as possible. To a certain extent, technology and available development time place bounds on what features may be implemented in any
10 given device. Thus, in the past, prior SPS devices have experienced drawbacks and limitations in areas, as examples, including receiver managers, signal measurements, bit synchronization techniques, integrity monitoring, operational mode switching, measurement interpolation, and hardware and software satellite signal tracking loops. Such drawbacks limit the performance of the device, the ease of use and robustness of the device, and have
15 an impact on sales of the device.

[006] Therefore, there is a need for overcoming the problems noted above, and other previously experienced.

SUMMARY

20 [007] Other systems, methods, features and advantages of the invention will be or will become apparent to one with skill in the art upon examination of the following figures and detailed description. It is intended that all such additional systems, methods,

features and advantages be included within this description, be within the scope of the invention, and be protected by the accompanying claims.

BRIEF DESCRIPTION OF THE FIGURES

5 [008] The components in the figures are not necessarily to scale, emphasis instead being placed upon illustrating the principles of the invention. In the figures, like reference numerals designate corresponding parts throughout the different views.

[009] FIG. 1 shows a time domain histogram 100 and a frequency domain histogram 102.

10 [010] FIG. 2 shows a performance curve of the frequency domain histogram approach assuming a small frequency offset.

[011] FIG. 3 shows a plot of time to acquire bit synchronization.

[012] FIG. 4 shows mean frequency error estimate plots.

[013] FIG. 5 shows standard deviation plots for frequency error estimates.

15 [014] FIG. 6 shows mean frequency error estimate plots.

[015] FIG. 7 shows standard deviation plots for frequency error estimates.

[016] FIG. 8 shows mean code phase error estimate plots.

[017] FIG. 9 shows standard deviation plots for code phase error.

[018] FIG. 10 shows mean estimate error plots.

20 [019] FIG. 11 shows standard deviation plots for estimate error.

[020] FIG. 12 shows a technique for signal interpolation.

[021] FIG. 13 illustrates a plot of mean code phase parabolic interpolation output.

[022] FIG. 14 illustrates a plot of standard deviation of code phase parabolic interpolation output.

[023] FIG. 15 illustrates a plot of mean errors in the code phase parabolic interpolation output.

5 [024] FIG. 16 illustrates a plot of standard deviation errors in the code phase parabolic interpolation output.

[025] FIG. 17 shows an uncertainty region for a user location.

[026] FIG. 18 shows satellite signal peak determination technique.

[027] FIG. 19 shows a block diagram of a tracking loop.

10 [028] FIG. 20 shows a block diagram of a tracking loop.

[029] FIG. 21 shows a synchronization technique.

[030] FIG. 22 shows a synchronization technique.

[031] FIG. 23 illustrates a control module relation diagram.

[032] FIG. 24 shows a block diagram of a control module.

15 [033] FIG. 25 shows a module architecture overview.

[034] FIG. 26 shows a task flow diagram.

[035] FIG. 27 shows a control module behavior diagram.

[036] FIG. 28 shows a flow diagram of interface manager processing.

[037] FIG. 29 shows a flow diagram of interface manager processing.

20 [038] FIG. 30 shows a flow diagram of interface manager processing.

[039] FIG. 31 shows a state machine for a search strategy manager.

[040] FIG. 32 illustrates a relation diagram.

- [041] FIG. 33 shows a block diagram of a control module.
- [042] FIG. 34 illustrates a block diagram of a message handler.
- [043] FIG. 35 shows an architecture view for a control module.
- [044] FIG. 36 illustrates a flow diagram of initialization steps.
- 5 [045] FIG. 37 shows a flow diagram of open procedure steps.
- [046] FIG. 38 shows a flow diagram of control module execution.
- [047] FIG. 39 shows an interaction diagram of interactions between functional blocks.
- [048] FIG. 40 shows an interaction diagram of interactions between functional
10 blocks.
- [049] FIG. 41 shows an interaction diagram of interactions between functional blocks.
- [050] FIG. 42 shows a block diagram of a satellite positioning system receiver having an expert system.
- 15 [051] FIG. 43 shows a system control and management process diagram of FIG. 42.
- [052] FIG. 44 shows a layered diagram of a satellite positioning system receiver of FIG. 42.
- [053] FIG. 45 is a flow diagram of the control of a satellite positioning system receiver of FIG. 42.
- 20 [054] FIG. 46 is a flow diagram depicting strong signal cancellation in a weak spread spectrum signal using crosscorrelation.

[055] FIG. 47 shows a signal-processing diagram for identifying and removing CW jamming signals where the cancelled signal of FIG. 42 may be the CW jamming signal.

[056] FIG. 48 shows a block diagram of electrical components of FIG. 43.

[057] FIG. 49 shows a flow diagram of the identifying and removing CW jamming
5 signals.

DETAILED DESCRIPTION

[058] The discussion below is directed to a hardware and software architecture that provides control and feature systems for satellite positioning systems (SPS). Specific
10 features of the architecture include, as examples, an Expert System receiver manager; a joint detection, carrier centering and bit sync acquisition subsystem; a zoom in, zoom out detection and interpolation subsystem; a multi-dimensional measurement interpolation subsystem; a subsystem for mode switching between a navigational signal with Synch and without Synch; and an autonomous integrity monitoring subsystem for a receiver.
15 The architecture and the control and feature systems described below are not limited to the precise implementations described, but may vary from system to system according to the particular needs or design constraints of those systems.

[059] In one implementation, the architecture obtains data bit synchronization for signals with a carrier to noise (C/N0) ratio at or lower than 21 dBHz. Two different
20 approaches are described below for resolving approximately 1 ms of ambiguity within a 20 ms data bit in the signal transmitted by a given satellite. In other words, the approaches accurately determine, within a 20 ms window, the time at which a data bit transition has occurred to accurately locate a bit transition in the transmitted signal. The

approaches include a time domain histogram approach 100 and a frequency domain histogram approach 102, illustrated in Figure 1.

[060] In the time domain histogram approach, the architecture creates a time domain histogram from time domain samples of the signal transmitted by a given satellite. In summary, the architecture sums samples taken at a pre-selected rate (e.g., 1 ms samples) over a moving window with a pre-selected length (e.g., 20 ms). Subsequently, twenty different hypothesis are postulated, one for each 1 ms shift of the moving window. A histogram with twenty bins (each corresponding to a different hypothesis) is then built by accumulating the sum of the linear envelope, ($\sqrt{I^2 + Q^2}$), over successive data bits. The accumulation results in bins in the histogram of differing magnitudes. The bin with the largest magnitude corresponds to the hypothesis that is closest to the true data bit transition.

[061] In one implementation, the architecture may then obtain a refinement of the estimate by performing, a multipoint interpolation on the bins. For example, the architecture may perform a 3 point interpolation using the largest bin and two adjacent bins, one on each side of the largest bin.

[062] In the frequency domain histogram approach 100, the architecture takes a moving window of pre-selected length (e.g., 20 ms). The window may include twenty (20) 1ms samples. The architecture applies a sample to each of twenty (20) inputs of a multi-point Fast Fourier Transform (FFT) circuit. As one example, the FFT circuitry 104 may determine a 32 point FFT. Subsequently, a pre-selected number (e.g., twenty (20)) different hypotheses are postulated, for example one hypothesis for each 1ms shift of the

moving window and twenty (20) corresponding FFT operations each corresponding to a unique hypothesis.

[063] The architecture may then build a two dimensional histogram. One axis or dimension of the histogram may correspond to the 32 FFT output bins, and the other axis may then correspond to the 20 hypotheses. The histogram may be built by accumulating the linear envelope, $(\sqrt{I^2 + Q^2})$, over successive data bits. The accumulation results in bins in the histogram of differing magnitudes. A bin may be a counter or a more complex structure, implemented in either hardware or software. The bin with the largest magnitude corresponds to the hypothesis that is closest to the true data bit transition and for the frequency that is closest to the input carrier frequency. Hence, a search across the frequency dimension gives the architecture the closest frequency. At that frequency, the architecture then searches the hypothesis axis for the best bit synchronization (bit transition) hypothesis.

[064] Simulation results are presented below to highlight the performance of the two approaches noted above. The simulations assume equally likely random data bits of +1/-1. The simulation runs over approximately 25,000 trials, with a statistical analysis set forth below. For each trial, a stopping condition was in place, and was chosen such that the accumulations occur for longer periods when the signal is weaker, and when the number of transitions is less.

[065] A time based stopping condition may be determined by accumulating the envelope of the difference between the present data bit and the previous data bit,

$(\sqrt{(I_1^2 - I_2^2) + (Q_1^2 - Q_2^2)})$, summing over all the hypotheses. Note that the difference is noise only, if there is no actual bit transition and proportional to twice the signal amplitude if there is a transition. The accumulations terminate when the accumulated difference reaches a preset threshold. At weak signal strengths, the signal amplitude is smaller and takes longer to reach the threshold and hence the simulation runs longer.

[066] A frequency based stopping condition may be determined by accumulating the envelope as noted above, but having the accumulation performed on the output of the frequency domain histogram. That is, the architecture accumulates the envelope of the difference between the present data bit and the previous data bit (over all frequency bins) and sums over all the hypotheses.

[067] For the results demonstrated below, the time based stopping condition may be employed for both time and frequency histogram approaches. In the simulations, the true bit transition is randomly generated anywhere in the range of 0-20ms. If the error between the estimate and the true transition is greater or equal to 0.5ms, an error is declared. The error statistics are obtained from a pre-selected number (e.g., 25,000) of trials. The number of transitions (and time to obtain bit synchronization) is also determined. In addition, a time out condition with a pre-selected duration (e.g., 8 seconds), checked with a time out counter, is employed to prevent the loops iterating indefinitely.

[068] Table 1, below, provides a comparison of the time domain and frequency domain histogram approaches assuming a known carrier frequency. The probability of

wrong detection of the bit transition may be used to compare and choose between the two algorithms for any particular implementation.

Table 1			
C/N0 (dBHz)	Avg. number of transitions	Probability of wrong detection	
		Time Histogram	Frequency Histogram
45	2.75	0.00308	0.00316
	3.26	0.00052	0.00028
	3.73	0.00004	0.00008
30	10.95	0.00188	0.00188
	21.04	0.00004	0.00004
22	70.7	0.00136	0.00136
21	75.3	0.00376	0.00464
20	79	0.01040	0.01020

5

[069] Table 2 shows the detection errors for frequency errors within a bin for the two algorithms

Table 2			
C/N0 (dBHz)	Frequency error (Hz)	Probability of wrong detection	
		Time Histogram	Frequency Histogram
22	0.0	0.00	0.00
	8.0	0.00012	0.00004
	15.0	0.00824	0.00796
	24.0	0.829	0.00
	32.0	1.00	0.00

10

[070] As can be seen from Table 1, when the carrier frequency is known, the performance of the two algorithms is similar. Also from Table 2, the performance of the two algorithms is similar for frequency errors within a bin. Note that the bin 0 may be centered at 0Hz and bin 1 may be centered at 31.25Hz. The differences at 24Hz and 32Hz are due to the fact that in the frequency domain histogram, these frequencies fall in the vicinity of the adjacent bin.

[071] One advantage of the frequency domain approach is that it the architecture may employ if as a joint frequency synchronization and bit synchronization. That is, the frequency domain algorithm, while providing the benefits of the time domain approach, also operates over multiple frequency trials in parallel. Figure 2 shows a plot of the performance curve of the frequency domain histogram approach for a small frequency offset (2 Hz), where the criteria for stopping is the time domain based threshold count. The same threshold value was used for all C/N0 and for all frequency offsets plotted in Figure 2.

[072] The performance curve 200 depicts the time to acquire bit synchronization across C/N0 for the case where there is a small frequency error. At 22dBHz, only 1 error was observed out of 25,000 trials. Thus the performance of the frequency domain histogram approach is similar to the time domain approach, across C/N0s for small frequency offsets when using the same stopping criterion.

[073] Figure 3 shows a plot 300 of the time to acquire bit synchronization across C/N0 for the case where the stopping criterion is based on the output of the frequency

domain histogram. Curve 302 shows the time to acquire bit synchronization across C/N0 for the case where there is a small frequency error of 2 Hz. Curve 304 of bit synchronization has the advantage of simultaneously performing a frequency estimation and bit synchronization. Note that the time domain approach employs a certain amount
5 of information regarding the frequency error to accurately to provide reliable bit synchronization (in a serial fashion). In the joint approach, however, the architecture may obtain an estimate of the carrier frequency along with the bit boundary in a parallel fashion.

[074] The architecture further includes interpolation and smoothing circuitry and
10 methods to that improve resolution of carrier frequency and code phase estimates for ranging signals transmitted by the SPS satellites that arrive in weak condition. In one implementation, the architecture employs discrete values of carrier Doppler and code phase, and the interpolation and smoothing techniques improve on the discrete values. For example, the interpolation and smoothing techniques may process the quantized
15 frequency and time bins prepared as noted above with regard to bit synchronization and acquisition in order to improve a carrier frequency determination and a time determination.

[075] The architecture may perform carrier frequency interpolation in different ways. For example, assuming seven (7) 1ms coherent samples are input to an eight (8)
20 point FFT (with one zero for the remaining input) and 3426 (6×571) times non-coherent integration results in a total time of 24 seconds and the FFT computes eight (8) bin magnitudes each of resolution 125Hz. Without interpolation, the bin with the maximum

magnitude would ordinarily be chosen, yielding a possible error in the range of -62.5 to 62.5Hz in the absence of binning errors. Binning errors, which happen at low C/N0s, may result in larger errors.

[076] In the analysis that leads to choosing a frequency interpolation technique, the
5 frequency error is swept across one bin and the estimate for each frequency error is obtained as the bin with the maximum magnitude. The architecture then adjusts the frequency estimate by using an interpolation to improve the estimate, for example, a multi-point (e.g., 3-point) parabolic interpolation. This interpolation may employ the maximum magnitude bin and the magnitude of the adjacent bin on each side of the
10 maximum.

[077] The peak position of a sampled quadratic can be located using the sampled peak and the two adjacent peaks. For a sampled quadratic function y , with sampled peak y_m and the true peak δ samples from m , the three samples about the peak are related by

15

$$\begin{aligned}y_{m-1} &= a(m-1-\delta)^2 + b \\y_m &= a(m-\delta)^2 + b \\y_{m+1} &= a(m+1-\delta)^2 + b\end{aligned}$$

[078] Setting $m = 0$ and solving for δ yields

20

$$\delta = \frac{(y_{m+1} - y_{m-1})}{(2y_m - y_{m+1} - y_{m-1})}$$

[079] and $m - \delta$ provides an accurate peak of the sampled quadratic.

[080] Figure 4 shows mean plots 400 of the mean of the frequency error estimate, including a plot 402 (C/N0 of 12 dBHz), a plot 404 (C/N0 of 15 dBHz), a plot 406 (C/N0 of 17 dBHz), and a plot 408 (C/N0 of 13 dBHz). Figure 5 shows standard deviation plots 500 of the standard deviation of the frequency error estimates, including a plot 502 (C/N0 of 12 dBHz), a plot 504 (C/N0 of 15 dBHz), a plot 506 (C/N0 of 17 dBHz), and a plot 508 (C/N0 of 13 dBHz). The plots in figures 4-5 are plotted for frequency errors within one bin. The results are similar across other bins.

[081] Figures 6 and 7 are similar to figures 4 and 5, but assume a 16 point FFT in which the seven (7) 1ms samples are padded with nine (9) zeros and input to a 16 point FFT. Figure 6 shows mean plots 600 including a plot 602 (for C/N0 of 17 dBHz), a plot 604 (for C/N0 of 15 dBHz), and a plot 606 (for C/N0 of 13 dBHz). Figure 7 shows standard deviation plots 700 including a plot 702 (for C/N0 of 17 dBHz), a plot 704 (for C/N0 of 15 dBHz), and a plot 706 (for C/N0 of 13 dBHz).

[082] Evaluating code phase interpolation may be performed, in one instance, assuming zero frequency error and a total range of +/- 1 chip. Thus, for 0.5 chip correlator spacing, there are five (5) code phase bins, each spaced 0.5 chips apart, i.e. (-1, 0.5, 0, 0.5, 1). For the other correlator spacings, a similar analysis may be performed.

[083] I and Q samples for each of the five (5) assumed time hypothesis may be generated by the following equations:

$$I = \sqrt{2CT / N_0} \rho(\tau - \tau_0) + x$$
$$Q = y$$

[084] In the evaluation simulations, the code phase error, τ_0 , may be swept across one bin, for example, -0.25 chips to 0.25 chips and the estimate for each error may be obtained by identifying the bin with the maximum magnitude.

[085] The architecture may then improve the code phase error using the three (3)
5 point parabolic interpolation explained above, using the maximum magnitude code phase bin (out of the five bins as explained above) and the magnitude of the adjacent bin on each side of the maximum. Consideration may also be taken to account for the correlation between noise samples for bins which are spaced less than a chip apart.

[086] Figures 8 and 9 plot the average and standard deviation of the code phase
10 error estimates respectively, against the actual code phase offset from 1000 trials. The plots in the figures are plotted for phase errors within one bin, with similar results across other bins. Figure 8 shows mean plots 800 including a plot 802 (for C/N0 or 12 dBHz), a plot 804 (for C/N0 of 15 dBHz), and a plot 806 (for C/N0 of 17 dBHz). Figure 9 shows standard deviation plots 900 including a plot 902 (for C/N0 or 12 dBHz), a plot 904 (for
15 C/N0 of 15 dBHz), and a plot 906 (for C/N0 of 17 dBHz).

[087] An alternate method of interpolation yields the results shown in Figures 10 and 11. In the alternate method, the architectures selects four bins from the five bins of the code phase search space, then performs a four (4) point FFT employing the four (4) bins. The FFT outputs are then zero padded to twice the size and an inverse FFT using
20 the eight (8) point FFT is then carried out. The peak is then estimated from the eight (8) point inverse FFT output. In one implementation, the architecture chooses the four out of

the five bins so that the maximum bin and a higher adjacent bin occupy the center of this four bin selection that may be implemented as an array in hardware or software.

[088] Figure 10 shows mean plots 1000 including a plot 1002 (for C/N0 or 12 dBHz), a plot 1004 (for C/N0 of 15 dBHz), and a plot 1006 (for C/N0 of 17 dBHz).

5 Figure 11 shows standard deviation plots 1100 including a plot 1102 (for C/N0 or 12 dBHz), a plot 1104 (for C/N0 of 15 dBHz), and a plot 1106 (for C/N0 of 17 dBHz).

[089] The above techniques can be generalized to any correlator spacing desired.

Figure 12 presents the generalized technique 1200. For instance, for a correlator spacing of $1/N$ chips (Step 1202), there would be a total of $2N+1$ bins to cover the range $[-1:1]$

10 chips (Step 1204). From these $2N+1$ bins, the architecture may select $2N$ bins as described above (Step 1206). The architecture may then perform a $2N$ FFT on these $2N$ bins (Step 1208), followed by a padding of $2N$ zeros to the FFT output (Step 1210), and then an $4N$ size inverse FFT (Step 1212).

[090] Table 3, below, shows the effects of binning errors for the cases considered
15 above (again assuming 1000 trials). Table 3 provides slightly pessimistic bounds for the case of frequency/code phase that lie at the edge of the bin, because in reality these scenarios will result in useful energy in the adjacent bins.

Table 3				
	Pfa for NCS = 571		Pfa for NCS = 6*571	
	$\Delta f = 0$	$\Delta f = \text{fbin_size}/2$	$\Delta f = 0$	$\Delta f = \text{fbin_size}/2$
Carrier doppler interpolation (8 pt. FFT)	17 dB : 0.000	17 dB : 0.002	17dB : 0.000	17 dB : 0.000
	15 dB : 0.003	15 dB : 0.051	15dB : 0.000	15 dB : 0.000
	12 dB : 0.162	12 dB : 0.296	12dB : 0.001	12 dB : 0.016
	$\Delta \tau = 0$	$\Delta \tau = -0.20$	$\Delta \tau = 0$	$\Delta \tau = -0.20 \text{ chip}$

		chip		
Code phase interpolation (0.5 chip spacing)	17dB : 0.000 15dB : 0.006 12dB : 0.165	17 dB : 0.184 15 dB : 0.296 12 dB : 0.504	17dB : 0.000 15dB : 0.000 12dB : 0.000	17 dB : 0.012 15 dB : 0.071 12 dB : 0.226

[091] Figure 13 and 14 plot the output of the code phase parabolic interpolation for a 12dBHz signal with and without binning errors for a Non-Coherent Sum (NCS) of 571.

5 Figure 13 shows mean plots 1300 including a plot 1302 that includes binning errors, and a plot 1304 that does not include binning errors. Figure 14 shows standard deviation plots 1400 including a plot 1402 that includes binning errors, and a plot 1404 that does not include binning errors.

[092] The effect of the limited bandwidth on the correlation function may be
10 estimated for the code phase parabolic interpolation. For example, assume a chip spacing of 1/8 chip, no binning errors, and parabolic interpolation, with the correlation triangle filtered by a 6MHz bandwidth filter and an unsynchronized decimator for the 1/8 chip spacing. Figure 15 shows mean plots 1500 of the mean of the errors with 1502 and without 1504 the filter. Figure 16 shows standard deviation plots 1600 of the standard
15 deviation of the errors with the filter 1602 and without the filter 1604. Near the peak of the correlation triangle, the variance from the filtered correlation triangle is higher due to the flattening in the triangle.

[093] In one implementation, employing Doppler frequency interpolation, the parabolic interpolation with padding of nine zeros may provide an improvement at weak
20 signal levels. For the code phase interpolation, the zero padded FFT algorithm provides

lower errors in the center of the bin compared to the parabolic interpolation and in larger variation in mean values.

[094] The architecture also performs peak assignment, for example, to choose the correct set of peaks (one for each satellite) from a given set of multiple peaks for the satellites. Figure 17 shows a uncertainty region 1700 that includes a reference position 1702 (x), a true user position 1704 (differing by delta x), and the i-th GPS Satellite 1706. The technique for peak assignment may operate on input data including aiding information with respect to an assumed reference position (e.g., the reference position 1702 at the center of the uncertainty region).

10 [095] The aiding information, as examples, may include a list of visible satellites (pseudo random noise (PRN) IDs), code phase indices (modulo 1023) for each satellite (e.g., at 1 chip resolution), Doppler values, line of sight (los) vectors to the satellites, a maximum horizontal position error (in meters, as shown by the radius of the circle in Fig. 17, and a maximum velocity error (in m/s).

15 [096] Equation 1 shows the measured data:

$$\begin{aligned} PRN_1 &= \{p_{11}, p_{12}, \dots, p_{1N}\} \\ PRN_2 &= \{p_{21}, p_{22}, \dots, p_{2N}\} \\ &\vdots \\ PRN_M &= \{p_{M1}, p_{M2}, \dots, p_{MN}\} \end{aligned}$$

Equation 1

[097] where there are M satellites and a set of N peaks for each satellite. Each peak is characterized by a corresponding code offset modulo 1023 (i.e., $0 \leq p_{ij} \leq 1022$), carrier frequency, and amplitude. In other words, each element in the above {M,N}

20

matrix is characterized by a code offset, frequency, and amplitude parameter. Thus, element P_{ij} will be characterized by 3 parameters $\{c_{p_{ij}}, d_{p_{ij}}, a_{p_{ij}}\}$ where $c_{p_{ij}}$ is the code phase index of P_{ij} , $d_{p_{ij}}$ is the Doppler of P_{ij} , and $a_{p_{ij}}$ is the amplitude of P_{ij} .

[098] In performing peak assignment, the architecture may assume that the peaks
5 are arranged in the order of decreasing amplitudes for a given satellite, that the satellites are arranged in descending order of their strengths (e.g., based on the first element for each satellite (i.e., each row)), and that aiding information is available for the PRN ids in the measured data (Equation 1).

[099] The first two assumptions together imply that the first row will correspond to
10 the strongest satellite and within the first row; the peaks are arranged in the descending amplitudes. Arranging the data in this manner may improve search speed, in the case where the architecture does not perform an exhaustive search of all possible combinations, while increasing the probability of finding the correct set of peaks.

[0100] The peak indices and peak Doppler values may be obtained through the
15 acquisition process (possibly aided). Hence, it is likely that the measured peak indices and Doppler values in Equation 1 lie within a window, bounded by position uncertainty, velocity uncertainty, time uncertainty, and frequency uncertainty.

[0101] The architecture, in one implementation, will determine a set of correct peaks according to criteria discussed below. The determined set of peaks (given by
20 $[p_{11} \ p_{21} \ \cdot \ \cdot \ \cdot \ p_{M1}]$) may be an array with M elements with each element corresponding to a unique satellite. The array of M elements may be implemented in

hardware or software as a data structure such as an array, link list, or other structure that maintains the relationship of the array elements.

[0102] In determining the set of correct peaks, the architecture may proceed according to the determination technique 1800 shown in Figure 18. The technique 1800 generally includes the steps of: Pruning, Upper Bounds, and Applying a Decision Technique. Pruning preprocess (Step 1802) the measured data to reduce the size of the data set (the number of peaks). In the Upper Bounds step, the architecture employs (Step 1804) the uncertainty information (position and time) and the LOS vectors to obtain bounds on the uncertainty between the measured index (Doppler) values and the reference index (Doppler) values. During application of the decision technique (Step 1806), the architecture applies a decision technique that employs the uncertainty bounds and the measured data to arrive at a determined set of peaks.

[0103] In the discussion below, reference to single differences are references to differences between satellite *i* with satellite *j* while double differences are the differences on single differences between a user's receiver and reference data (e.g., the aiding information).

[0104] In the pruning step, the architecture reduces the size of the measured data, while employing little or minimal processing. In one implementation, the architecture performs pruning by employing the amplitude information (recall that peaks are arranged in order of decreasing amplitudes).

[0105] For example, the architecture may discard all peaks that satisfy:

[0106]
$$a_{p_i} < k_1 * a_{p_{i1}}$$

[0107] where i in the above equation is the satellite number and $j = 2, 3, \dots, 8$ denotes the position in the set.

[0108] $k_1, (0 < k_1 < 1)$

[0109] is a threshold constant and thus, if

5 [0110] $k_1 = 0.5$

[0111] the architecture discards peaks which are less than half the size of the strongest peak. For the satellites with the strong signals, where a dominant peak stands out, a set with one element per strong satellite may result.

[0112] In the step of applying upper bounds, the architecture employs apriori
10 uncertainty information on the position and velocity to obtain upper bounds on the expected code phase index (Doppler) difference between the values provided at the reference and those measured at the true position.

[0113] With regard again to Figure 17, that figure shows the relationship between a satellite i and an assumed approximate user position x 1702. The true position 1704 of
15 the user "u", is unknown. The aiding information provides an upper bound on the difference between the assumed position 1702 and the true position 1704.

[0114] The range measured by the user from the true position at time t to satellite i is given by:

$$r_i(t) = \hat{r}_i(t) - \hat{l}_i(t) * \Delta x + c * b_u(t) + v_i$$

20

Equation 2

[0115] where c is the speed of light (m/s), b_u is the bias in the receiver's clock (s), The term $v_i(t)$ represents the measurement noise (m). The terms with $\hat{}$ denote the estimate values (at the reference). The line of sight vectors are given by

$$\hat{l}_i(t) = \frac{s_i(t) - x}{|s_i(t) - x|}$$

[0116]

5 [0117] Note that in the equation 2 above, $r_i(t)$ denotes the range measurement at the true user position u . The first term on the right side $\hat{r}_i(t)$ represents the range measurement at the center of the uncertainty (reference position). The second term denotes the error due to the uncertainty in true receiver position and the third term denotes the bias in the receiver time.

10 [0118] Calculating the single differences from two different satellites, i and j :

$$r_i(t) - r_j(t) = (\hat{r}_i(t) - \hat{r}_j(t)) - (\hat{l}_i(t) - \hat{l}_j(t)) * \Delta x + (v_i - v_j)$$

Equation 3

[0119] In the Equation 3, the left hand side denotes the single difference in ranges
15 between satellites i and j as referenced to the true user position u . The first difference term on the right hand side denotes the range differences between satellites i and satellite j at the center of the uncertainty. The second term represents the error due to the user position uncertainty. Note that this is also a function of the geometry of the satellites.

20 [0120] Rewriting equation 3 to express the double differences and omitting the measurement noise term gives:

$$[r_i(t) - r_j(t)] - [(\hat{r}_i(t) - \hat{r}_j(t))] = (\hat{l}_i(t) - \hat{l}_j(t)) * \Delta x$$

Equation 4

[0121] Similarly for Doppler:

$$[d_i(t) - d_j(t)] - [(\hat{d}_i(t) - \hat{d}_j(t))] = (\hat{l}_i(t) - \hat{l}_j(t)) * \Delta u$$

Equation 5

[0122] where d_i and \hat{d}_i are the measured Doppler at the true user location and the reference location respectively due to satellite i , and Δu denotes the uncertainty in user velocity.

[0123] Equations 4 and 5 provide the architecture with upper bounds on the double differences (between satellite i and satellite j) in code phase indices (Doppler) between those at the reference position and those measured at the true position.

[0124] Next, the architecture applies a decision technique to determine a selected peak for each satellite from the set of peaks obtained at noted above. In one implementation, the architecture employs a cost vector in arriving at a determined set of peaks. Thus, for example, the architecture may select a set of peaks from the matrix in Equation 1 by forming a column vector (one column), where each element in the column vector corresponds to a unique satellite.

[0125] For instance, choosing the first elements for each satellite yields the vector:

$$[p_{11} \quad p_{21} \quad \cdot \quad \cdot \quad \cdot \quad p_{M1}]$$

[0126] For the chosen column vector, the next step is to form the single differences in their code phase indices and Doppler. For the amplitudes, the architecture may form the corresponding pairwise product of the amplitudes:

$$\begin{aligned} &[(c_{p_{11}} - c_{p_{21}}) \ (c_{p_{11}} - c_{p_{31}}) \ \dots \ (c_{p_{11}} - c_{p_{M1}}) \ (c_{p_{21}} - c_{p_{31}}) \ \dots \ (c_{p_{21}} - c_{p_{M1}}) \ \dots \ (c_{p_{M-11}} - c_{p_{M1}})] \\ &[(d_{p_{11}} - d_{p_{21}}) \ (d_{p_{11}} - d_{p_{31}}) \ \dots \ (d_{p_{11}} - d_{p_{M1}}) \ (d_{p_{21}} - d_{p_{31}}) \ \dots \ (d_{p_{21}} - d_{p_{M1}}) \ \dots \ (d_{p_{M-11}} - d_{p_{M1}})] \\ &[(a_{p_{11}} * a_{p_{21}}) \ (a_{p_{11}} * a_{p_{31}}) \ \dots \ (a_{p_{11}} * a_{p_{M1}}) \ (a_{p_{21}} * a_{p_{31}}) \ \dots \ (a_{p_{21}} * a_{p_{M1}}) \ \dots \ (a_{p_{M-11}} * a_{p_{M1}})] \end{aligned}$$

5

[0127] The architecture may employ the absolute values of these terms. For the code phase indices, the architecture may employ:

$$\begin{aligned} &|c_{p_{11}} - c_{p_{21}}| \quad \text{if } |c_{p_{11}} - c_{p_{21}}| < 512 \\ &1022 - |c_{p_{11}} - c_{p_{21}}| \quad \text{otherwise} \end{aligned}$$

10

[0128] Note that the size M of the single difference vectors above is M=2. Thus for M=5, there are a total of 10 elements in each of the vectors above. The architecture repeats the above step for the estimates at the reference position 1702. Thus for the given code phase indices (Doppler) at the reference position (a vector of size M), the architecture forms the single differences. In addition, the architecture also forms the magnitude of the single differences for the line of sight vectors. All the resulting vectors are of size M=2 in the current implementation.

15

[0129] The architecture then, by employing the results of the bounding steps, obtains the upper bound in the error differences (double differences) between the values at the true position 1704 and the reference position 1702 (right hand sides of equations 4 and 5).

20

[0130] For the code phase indices, the bound will be: position uncertainty (in chips) * LOS vectors (magnitude of single differences).

[0131] For Doppler values, the bound will be: (velocity + position) uncertainty (in Hz) * LOS vectors (magnitude of single differences).

25

[0132] The architecture also obtains the (double difference) error term for code phase indices and Doppler. The error term is the difference in the single difference values at

the true position (explained above) and those at the reference position (explained above).

Note that the error term vector is also of size $M=2$.

[0133] Next, the architecture compares the error terms against the bounds on an element by element basis. If an element of the error term is greater than the corresponding bound element, the architecture increases the cost vector proportional to the inverse of the peak amplitude pairwise products formed as noted above and proportional to the difference in the error terms (double differences) and the upper bound. Note that this weight will be assigned to both elements (i.e. peaks) that were used in forming the single difference. Then, if the error term is less than the corresponding bound, the cost vector is not changed. The architecture may follow this procedure for all M choose 2 elements. At the end of this step, the architecture obtains a cost vector of size M .

[0134] The architecture may then repeat the same procedure for the Doppler terms without resetting the cost vector. When the cost vector is equal to zero (all M elements identically zero), the architecture may determine that this corresponds to the optimum peak vector, and stop the search. Otherwise, the architecture saves the cost vector, resets it and returns to choose a new set of peaks as noted above with regard to forming the column vector.

[0135] When all combinations of peaks have been searched without having a zero cost vector, then the architecture may select the set of peaks with the lowest cost vector magnitude. In the case of a tie, the architecture may select the set of peaks that occurs, for example, first in the search process.

[0136] The discussion below details the tracking system for the architecture for strong and medium signal operation. The following abbreviations may be used below:

Alpha, Beta: Generic filter coefficients that may take different values at different instances; FFT: Fast Fourier Transform; SPS: Satellite Positioning System; HWTL:

5 Hardware Tracking Loop; NCO: Numerically Controlled Oscillator; PDI: Pre Detection Integration; RAM: Random Access Memory; S_Gain: Filtered Signal amplitude estimate employed to normalize the tracking loops; SWTL: Software Tracking Loop; T1: Basic Time epoch for Subsystem 2; Threshold, Threshold1, Threshold2: Generic Threshold values that may take different values at different times.

10 [0137] Figure 19 shows a block diagram of a tracking loop 1900 employed by the architecture. In particular, figure 19 shows a hardware tracking layer 1902 in communication with a subsystem labeled Subsystem 2 1904. The Subsystem 2 1904 is in communication with the Subsystem 3 1906, the NCS Buffer 1908, and the software tracking loop 1910. The subsystem 2 1904 may include a matched filter and associated
15 logic/circuitry, for example, while the Subsystem 3 1906 may include a FFT circuit and associated logic/circuitry and non coherent sum (NCS) accumulators and associated logic/circuitry.

[0138] The hardware tracking layer 1902 operates at PDI rate. In one implementation, the mode transition to activate HWTL may be under software control at
20 the successful completion of the Acquisition process. The software for the software control may be written in a high level language such as "C", "C++", machine language, or even an interpreted language such as "PERL".

[0139] Inputs from the hardware include: I_e , Q_e and I_l , Q_l , f_{+e} and f_{-e} , f_{+l} and f_{-l} for early and late taps, where, I_e = In Phase Correlation for early tap, Q_e = Quad Phase Correlation for early tap, I_l = In Phase Correlation for late tap, Q_l = Quad Phase Correlation for late tap, f_{+e} = Correlation Magnitude for +1 frequency bin offset for early tap, f_{-e} = Correlation Magnitude for -1 frequency bin offset for early tap, f_{+l} = Correlation Magnitude for +1 frequency bin offset for late tap, and f_{-l} = Correlation Magnitude for -1 frequency bin offset for late tap. The values may be autoscaled from hardware for normalization purposes.

[0140] Using information from the Acquisition process, the architecture may initialize the following states: 1) Code Phase, 2) Carrier Frequency, 3) Carrier Phase, and 4) S_Gain (if an amplitude estimate is available).

[0141] The architecture employs the following discriminator functions:

[0142] Code Discriminator:

$$D = |e| - |l|$$
$$|e| = \sqrt{I_e^2 + Q_e^2}$$
$$|l| = \sqrt{I_l^2 + Q_l^2}$$

[0143] Carrier Phase Discriminator:

$$\phi = \text{sign}(I_p) * Q_p$$
$$I_p = I_e + I_l$$
$$Q_p = Q_e + Q_l$$

[0144] Carrier Frequency Discriminator (using magnitude or magnitude approximations):

$$\delta f = f_+ - f_-$$

$$f_+ = f_{+e} + f_{+l}$$

$$f_- = f_{-e} + f_{-l}$$

- 5 [0145] The subsystem 2 1904 may perform the following determinations:

Carrier Phase += Carrier Frequency

Carrier Frequency += Carrier Frequency Rate

Code Phase += Delta Carrier Phase

- 10 [0146] The Tracking Loop Equations are given below:

Carrier Phase += $K1 * \phi + K2 * \delta f + \text{Aid 1}$

Carrier Frequency += $K3 * \phi + K4 * \delta f + \text{Aid 2}$

Carrier Frequency Rate += $K5 * \phi + K6 * \delta f + \text{Aid 3}$

Code Phase += $K7 * D + \text{Aid 4}$

15

[0147] Where the gains K1 through K7 are input from software, for example, at 100 ms rates. The HWTL mode transitions are controlled through the values of these gains. In one implementation, Aid 1 may be carrier phase information obtained outside the tracking loop, Aid 2 may be carrier frequency information obtained outside the tracking loop, and Aid 3 may be carrier frequency rate information obtained from outside the tracking loop. The Aid 1 through Aid 3 information may be obtained, for example, from other subsystems in the architecture, e.g., from the receiver or external to the receiver, as from a Navigation subsystem, inertial sensors.

20

[0148] The tracking modes and transition between modes for one implementation are explained below, as controlled through software at a 100 ms rate by setting the gains K1-K7.

25

[0149] Initialization Mode: The architecture initializes the tracking loops with a wide bandwidth (e.g., 1/10 of iteration rate PDI) frequency loop. The code loop is initialized with wide bandwidth (e.g., 5 Hz * Acquisition Error Estimate). A transition may be made to narrow bandwidth frequency loop if the filtered frequency error estimate less than threshold1. Transition to a wide carrier phase loop may be made if the filtered frequency error estimate is less than threshold2. The PDI may be set to, for example, 4ms, and the FFTs employed may be eight (8) point FFTs.

[0150] Narrow Frequency Loop: On entry from Initialization Mode, the tracking loops are set to narrow bandwidth frequency loop (e.g., 1/20 of PDI rate). Narrow code loop (recommended bandwidth 1/3 Hz) is employed if the estimated code error is less than 0.1 chips. On the other hand, a transition to carrier phase loop may occur if the filtered frequency error estimate less than threshold1. A transition back to Initialization Mode may be made if the filtered frequency error estimate greater than threshold2. The PDI may be set to, for example, 4ms, and the FFTs employed may be eight (8) point FFTs.

[0151] Wide Carrier Phase Loop: Entry may be made from Initialization Mode or Narrow Frequency Loop mode. Here, the tracking loops are set to wide carrier (e.g., bandwidth ¼ of PDI rate) and narrow frequency combined loop using K1-K7. Narrow code loop (e.g., bandwidth 1/3 Hz) is employed if the estimated code error less than 0.1 chips. On the other hand, a transition to Narrow Frequency Loop may be made if 1) the carrier phase lock is lost and 2) the filtered frequency error estimate less than threshold1. A transition to Initialization Mode may be made if 1) the carrier phase lock is lost and 2)

the filtered frequency error estimate less than threshold2. Also, a transition to Narrow Carrier Loop may be made if filtered carrier phase error less than threshold. For the wide carrier phase loop, the PDI may be 4ms in absence of bit synchronization and 20ms in presence of bit synchronization, and the FFT may be a twenty (20) point FFT.

- 5 [0152] Narrow Carrier Phase Loop: Entry may be made from Wide Carrier Phase Loop. The tracking loop may be set to narrow carrier (e.g., bandwidth 1/10 of PDI rate) and narrow frequency combined loop using K1-K7. A narrow code loop (e.g., bandwidth 1/3 Hz) may be employed when the estimated code error is less than 0.1 chips, with a transition to Narrow Frequency Loop if 1) carrier phase lock is lost and 2) the filtered
- 10 frequency error estimate less than threshold1. A transition to Initialization Mode may be made when 1) carrier phase lock is lost and 2) filtered frequency error estimate less than threshold2. Here, the PDI may be set to 4 ms in absence of bit synchronization and 20ms in presence of bit synchronization, and the FFT may be a twenty (20) point FFT.

- [0153] Table 4 summarizes a Mode Transition Table in which the columns are the
- 15 source modes and rows are the destination modes. The entries in the table describe the transition condition.

Table 4				
	Init	Narrow Freq	Wide Carrier	Narrow Carrier
Init	X	filtered frequency error estimate < threshold1	filtered frequency error estimate < threshold2.	X
Narrow Freq	filtered frequency	X	filtered frequency	X

	error estimate > threshold2		error estimate < threshold1	
Wide Carrier	Mode if (carrier Phase lock lost and filtered frequency error estimate > threshold2).	if (carrier Phase lock lost and filtered frequency error estimate > threshold1).	X	X
Narrow Carrier	if (carrier Phase lock lost and filtered frequency error estimate > threshold2).	if (carrier Phase lock lost and filtered frequency error estimate > threshold1	X	X

[0154] Similarly, Table 5 shows a Code Loop Transition Table.

Table 5		
	Init (High Bandwidth)	Low Bandwidth
Init (High Bandwidth)	X	estimated code error < 0.1 chips
Low Bandwidth	estimated code error > 0.1 chips	X

[0155] In one embodiment, each of the modes will transition out of the tracking
5 system if code and frequency lock are lost. At that point, the acquisition process will
commence or re-commence.

[0156] Hardware Update. At the end of processing for the Subsystem 3, the Carrier
Phase, Carrier Frequency, Carrier Frequency Rate and Code Phase is updated in the
Subsystem 2 state. At this time the subsystem 2 is typically inactive. The code state has
10 a coarse resolution and the code phase equations can accumulate small code phase

updates. When the code lsb is reached the code state in subsystem 2 will be updated.

This can be accomplished as shown in Figure 20.

[0157] Figure 20 shows a tracking loop 2000 including delay circuits 2002 and 2004, alpha multiplier circuits 2006 and 2008, and beta multiplier circuits 2010 and 2012. Also shown are the constant multiplier 2014, a comparison circuit 2016, and summers 2018 and 2020.

[0158] The tracking loop 2000 describes an approximation for the normalized early – late triggering the changes to the code state. N is the number of least significant bits (LSBs) in the code state to change.

10 [0159] AAGC Normalization / Autoscale may be given by:

$$S_Gain(t+1) = Alpha * S_Gain + Beta * (|Ip| + |Qp|)$$

[0160] Where the S_Gain is employed to normalize the loop coefficients and "t" is the time index. Note that S_Gain may also be employed to Normalize the early minus late output to determine the update to the code state in hardware.

15 [0161] Software Tracking Loop: The SWTL may be activated at the same time as the HWTL by software control at the end of a successful acquisition process. The SWTL is operated, for example, at a 100 ms rate. In one implementation, the SWTL includes buffered reports at the PDI rate and reports on NCO states. The software may compute more accurate error estimates use aiding information from a Navigation process and optionally outside sources in order to construct a correction approximately every 100 ms to adjust the HWTL. In certain cases the HWTL may also be disabled and the SWTL may operate autonomously, generating corrections to the hardware NCOs at rates

different than 100 ms. The hardware may then provide interrupts at higher rates and the SWTL will operate at rates faster than 100 ms in such cases. The HWTL may be disabled using a control bit for each channel independently.

[0162] Hardware Inputs. The hardware provides input to the NCS Buffer and Track
5 History (TH) Buffer at a 100ms rate, for example. The input may include I and Q correlation outputs at each PDI for various offsets, noise sums for PDIs, AutoScale Values, NCO states sampled every PDI, Time Marks representing time of Measurement Report, and so forth.

[0163] State Initialization. The following states may be initialized using information
10 from the Acquisition phase: a) Code Phase, b) Carrier Frequency, c) Carrier Phase, d) S_Gain (when, for example, an amplitude estimate is available).

[0164] The code discriminator may be given by the following equations:

$$\begin{aligned} D &= |E| - |L| \\ |E| &= \text{Alpha} * |Ee| + \text{Beta} * |Ee+| \\ 15 \quad |L| &= \text{Alpha} * |Le| + \text{Beta} * |Le+| \end{aligned}$$

[0165] where Ee and Le are the early and late I and Q values for one tap off from prompt and Ee+ and Le+ are the early and late I and Q values for two tap off from prompt, and:

$$20 \quad \text{Filtered Code Error} = \text{Alpha} * \text{Filtered Code Error} + \text{Beta} * D$$

[0166] The carrier phase discriminator may be given by:

$$\phi = \arctan (Q_p, I_p)$$

[0167] Where ϕ is computed for each PDI and a $d\phi$ is maintained in software.

[0168] The carrier frequency discriminator may be given by:

5
$$\delta f = \text{Alpha} * (f_t - f_c) + \text{Beta} * (I_p(t+1) * Q_p(t) - I_p(t) * Q_p(t+1))$$

[0169] where t is the time index.

[0170] The architecture may iterate the tracking equations at a pre-selected rate, for example, at the PDI rate, with corrections back into the hardware each 100 ms, for
10 example.

[0171] The phase tracking loop equations may be given by:

15
$$\begin{aligned} \text{Carrier Phase} &+= \text{Carrier Frequency} + K1 * d\phi + \text{Aid 1} \\ \text{Carrier Frequency} &+= \text{Carrier Frequency Rate} + K3 * d\phi + \text{Aid 2} \\ \text{Carrier Frequency Rate} &+= K5 * \phi + \text{Aid 3} \end{aligned}$$

[0172] The frequency tracking loop equations may be given by:

20
$$\begin{aligned} \text{Carrier Frequency} &+= \text{Carrier Frequency Rate} + K4 * \delta f + \text{Aid 2} \\ \text{Carrier Frequency Rate} &+= K6 * \delta f + \text{Aid 3} \end{aligned}$$

[0173] The code loop equations may be given by:

Code Phase += Scale*(Carrier Phase (t) – Carrier Phase(t-1)) K7 * D +
Aid4

K7 initialized to Kmax (high bandwidth value) and updated as follows

$$K7(t+1) = \text{Alpha} * K7 + \text{Beta} * \text{Code Error Estimate}$$

5

If $K7 > K_{\text{max}}$ $K7 = K_{\text{max}}$

If $K7 < K_{\text{min}}$ $K7 = K_{\text{min}}$

[0174] Note that the tracking loop mode transition may be controlled by setting the gains K1 through K7. For AGC normalization, the following may apply:

10

$$S_Gain(t+1) = S_Gain_Rate * \text{Gamma} + \text{Alpha} * S_Gain + \text{Beta} * (|Ip| + |Qp|)$$

$$S_Gain_Rate(t+1) = \text{Alpha} * S_Gain_Rate + \text{Beta} * [(|Ip(t)| + |Qp(t)|) - (|Ip(t-1)| + |Qp(t-1)|)]$$

15

[0175] where, the S_Gain and the S_Gain Rate represent the estimate of magnitude of the signal and the rate of magnitude change for the signal. The S_Gain parameter may be used to normalize the tracking loop equation gains.

[0176] In one implementation, the architecture also employs Loss of Lock Detectors.

For example, for Code Lock, the architecture may determine the Signal to Noise Ratio at

20

rate according to:

$$\text{Noise Power} = \text{Sum}(In * In + Qn * Qn)$$

[0177] where the sum is carried over, for example, a 100 ms time period and In and Qn are I and Q outputs from a noise sum in the hardware reports at the PDI rate, and:

25

$$\text{Signal Power} = \text{Sum}(Ip * Ip + Qp * Qp)$$

[0178] where the sum is carried over, for example, a 100 ms time period, and:

Signal to Noise Ratio = Signal Power / Noise Power

Filtered SNR = Alpha * Filtered SNR + Beta * Signal to Noise Ratio +
Gamma * SNR_Rate

5 SNR_Rate = Alpha * SNR_Rate + Beta * [Signal to Noise Ratio (t) -
Signal to Noise Ratio (t-1)]

[0179] Then, when Filtered SNR less than the Threshold, then Loss Of Lock has
occurred.

[0180] For carrier phase lock, the architecture may estimate the filtered phase error
10 using a two quadrant arctan function computed every PDI, for example:

$$\phi = \arctan (Q_p, I_p)$$

$$\text{Filtered } \phi = \text{Alpha} * \text{Filtered } \phi + \text{Beta} * \phi$$

15 [0181] Then, when Filtered ϕ is greater than Threshold, then declare loss of lock.

[0182] For carrier frequency lock, the architecture may estimate filtered frequency
error:

$$\delta f = \text{Alpha} * (f_+ - f_-) + \text{Beta} * (I_p(t+1) * Q_p(t) - I_p(t) * Q_p(t+1))$$

20 [0183] where δf is computed at the PDI Rate, and:

$$\text{Filtered } \delta f = \text{Alpha} * \text{Filtered } \delta f + \text{Beta} * \delta f$$

[0184] Then, when Filtered $\delta f < \text{Threshold}$, then declare loss of lock.

[0185] Updates to the hardware may be in the form of Aid to the HWTL equations,
25 for example, every 100 ms as input from the software. The architecture may determine
the Aid according to:

software to hardware Aid = Software Estimate – Hardware Estimate

[0186] When the HWTL is disabled, the hardware Estimate = 0 for the phase and frequency and rates. The hardware will implement the aid when, for example, the software writes into pre-selected hardware registers.

[0187] In one implementation, the bit synchronization operates, for example, at 20-100 ms rates, and may be implemented in software.

[0188] The inputs to the bit synchronization process may include, from hardware, 20 ms accumulations (PDI) for 20 offsets (programmable). The Input Array histogram[20] may then include 20 ms PDI power accumulations.

[0189] The Histogram may be accumulated using the following equation for each satellite (SV):

$$\text{AccumHistogram}[20] += \text{Histogram}[20]$$

[0190] When offset information is available, then accumulate Histograms from multiple satellites employing the following equation:

$$\text{Accum HistogramMultiple SV}[20] += \text{AccumHistogram}[20 + \text{SV Offset}]$$

[0191] Then, the architecture may employ the following synchronization technique 2100 shown in Figure 21 to output a bit location for synchronization. First, the histogram peak and second peak are detected (Step 2102). Then, if ((Histogram Peak > Threshold) && (Histogram Peak – Histogram Second Peak) > Threshold) (Step 2104), then the bit

synchronization is complete, and the bit location is output (Step 2106). Otherwise, accumulation is continued (Step 2108).

[0192] An alternative approach may employ the synchronization technique 2200 shown in Figure 22 to output a bit location for synchronization. First, the architecture determines a cost function (Step 2202) by correlating a triangle, for example, 20 ms wide to the accumulated histogram. Then, $\text{Cost Function}(\text{offset}) = \text{Sum}(\text{triangle}(\text{ms}) * \text{Accum Histogram}(\text{ms} - \text{offset}))$ may be determined (Step 2204), with the sum carried over the 20 values. The maximum for the cost function(offset) may then provide the bit synchronization offset. Then, if $[\text{max}(\text{cost function}(\text{offset}) > \text{Threshold}) \&\& \{\text{max}(\text{cost function}(\text{offset}) - \text{second max}(\text{cost function}(\text{offset}))\} > \text{Threshold}]$ (Step 2206), then declare bit synchronization success (Step 2208).

[0193] Data demodulation may operate, for example, at 20-100ms rates, with an implementation in software. The demodulation may accept, as input, after bit synchronization success, I_p and Q_p for 20ms aligned to bit boundaries. The implementation may proceed according to: If carrier Phase Lock, then $\text{Data Bit} = \text{sign}(I_p)$; else $\text{Data Bit} = \text{sign}[(I_p + j * Q_p) * \exp(j * \text{Carrier Phase Error Estimate})]$, where j is the complex coefficient. This approach is a more general case of differential decoding when the carrier phase estimate error is generated from the previous bit.

[0194] Frame synchronization may also operate, for example, at 20-100ms rates, with an implementation in software. The frame synchronization may operate on a bit stream after data demodulation provided as a array of binary values per SV.

[0195] The frame synchronization may proceed according to: Cold Start - Preamble Synchronization. Identify Preamble in the bit pattern. Decode pre-determined Hand-Over-Word (HOW). Identify second preamble 6 seconds later and identify second HOW off by 1. When an approximate time is available, the frame synchronization may proceed according to: 1) Identify preamble. Decode HOW. If HOW is within time uncertainty, then declare frame synch complete; 2) Verify Frame Synch - Repeat the Identify process and confirm HOW change by 1 with 6 second offset.

[0196] When Aiding Information Available and 1 SV completed Frame Synch, then: determine pre positioning for the SVs needing frame synch using time from the first SV frame synch; determine frame starts for the SVs of interest relative to local time; and set the frame synch information. When Aiding Bits are available, then determine a cost function using the aiding bits available: $\text{Cost Function}(\text{offset}) = \text{Sum}(\text{Aiding Bits}(\text{bit number}) * \text{Bit Stream}(\text{bit number} + \text{offset}))$, with the Sum carried over, for example, all the available aiding bits. Then, if the SV offset information is available combine the cost function for multiple SVs using the following equation:

Combined Cost Function (offset) += Cost Function(offset + SV offset)

If [max((Combined Cost Function(offset) > Threshold) && (max(Combined Cost Function(offset)) - second max(Combined Cost Function(offset)) > Threshold)], then declare frame synch Complete.

[0197] In one example implementation, the architecture may include a control module that is responsible for acquisition and tracking of SPS satellites. An exemplary module is discussed below. The following abbreviations may be used: Rx - receiver,

NVM - non volatile memory, TTFF - time to first fix, and ATX - acquisition track cross-correlation.

[0198] The control module may interpret commands from the a Control Manager, as well as determine/handle/implement search strategies and determine hardware allocation.

5 Figure 23 shows a relation diagram 2300 that, in one implementation, illustrates the interactions between the ATX control module and the rest of the SPS system. Note that the relation diagram 2300 includes interaction between the module of interest and other modules (actors). Interactions may be of several types including: uses (calls), creates, updates and obeys. The relation diagram 2300 identifies modules and runtime elements
10 in an overall reception architecture.

[0199] Table 6 explains the relationships shown in Figure 23.

Table 6
Reset: Commands the module to reset, generally as part of a GPS Start/Stop sequence. Startup: Commands the module to initialize (Open command). ATX Control Manager: Sends SV records to ATX Control and requests status/measurement data. ATX Task: Requests that ATX Control responds to commands (100 ms interval) DSP: Hardware interface functions. For information only. Xcorr: Responds to cross correlation records from ATX Control. Provides measurements/cross correlation status to ATX Control. Tracking: Responds to tracking records from ATX Control. Provides measurements/tracking status to ATX Control. Acquisition: Responds to acquisition records from ATX Control. Provides acquisition status to ATX Control.

[0200] Table 7 shows exemplary queries and commands that the ATX Control module may support.

Table 7 Commands and Queries for ATX Control	
Command/Query	Description
Open request	Request to open the module, will also force initialization of module. Will initialize the Acquisition, Track, Xcorr, and DSP Manager modules also.
Close request	Request to close the module. Will close the Acquisition, Track, Xcorr, and DSP Manager modules also.
Reset request	Request to close and open the module.
Memory mode request	Request to change the DSP memory map.
Support satellite acquisition and track command	ATX Control module will process the commands from ATX Control Manager.
Support request for acquisition and track status/measurements	ATX Control module will respond to request for current status of ATX subsystem for the ATX Control Manager module.
Support query of DSP module	ATX Control module will query the DSP module for status (memory available, execution timeline).
Create acquisition command	Command the Acquisition module to perform the acquisition command.
Request acquisition status	Query Acquisition module for acquisition status.
Create tracking command	Command the Tracking module to perform the track command.
Request tracking status & measurements	Query Tracking module for tracking status and measurements.

Create cross correlation command	Command the Xcorr module to perform the track command.
Request cross correlation status	Query Xcorr module for cross correlation status.

[0201] Table 8 shows features of the ATX control module, in one implementation.

Table 8	
Strategy	Action
Separate Generic GPS Rx Modules from hardware dependent GPS modules. Use a single interface to connect the two groups of modules.	Thus, in one implementation, there exists an interface to communicate to the HI modules.
Ensure that initialization is explicit and ensure every module has the capability to initialize itself on demand.	Thus, initialization blocks may be provided.
Isolate the Core functionality in a module from its interfaces.	Thus, the acquisition and track strategy logic may be isolated from external interfaces by creating interface blocks.

5

[0202] In one implementation, the ATX Control module interacts with hardware, but does not require NVM space. The communication between functional blocks may be accomplished through software protocols (denoted with a capital P): PatxCom – ATX Control command interface; PatxReq – ATX Control request interface; Pacq – Interface to Acquisition Module; Ptrack – Interface to Track Module; Pxcorr – Interface to Xcorr
10 Module; Pdsp – Interface to DSP Module.

[0203] A protocol may be defined (including handshaking) from the point of view of a particular functional block. Other functional blocks that wish to communicate may use the opposite or conjugate protocol indicated by the suffix "-conj". Components (functional blocks) are denoted with a starting capital "C". Note that the functional blocks are presented in such a way to indicate compliance with the strategies and the specific product requirements and are implementation independent. In the module architecture, certain conceptual elements may be lumped together in a single module to simplify the implementation.

[0204] Figure 24 presents a block diagram 2400 of the ATX control module, including the components 2402-2420 explained below.

[0205] The protocols used by the ATX control functional blocks are shown in Figure 24, as PatxCom and PatxReq. In one implementation, the ATX Control protocols do not perform handshaking. The discussion below sets forth the types of messages, the expected sequence, and the arguments.

Table 9	
<<Protocol>> .	
PatxCom	
Incoming	
ATXCom(command type, pointer to command data, success/failure)	
Outgoing	
....	
Sequencing (CatxComInt, into ←, out of →)	
Direction	
←	ATXCom

Table 10	
<<Protocol>> PatxReq	
Incoming ATXReq(data requested, pointer to copy data, success/failure)	
Outgoing	
Sequencing (CatxReqInt, into ←, out of →)	
Direction	
←	ATXReq

[0206] In one implementation, the ATX Control module communicates across three layers: it communicates with the ATX Control Manager in the Applications layer and
5 with the DSP Manager and hardware Timer modules in the Hardware/Driver layer.

[0207] Also, the ATX Control module communicates between tasks. It runs from the ATX task and communicates with the ATX Control Manager in the Rx Controller task. The DSP Manager runs from an ISR interrupt.

Table 11			
Component and Protocol Elements		Module Elements	
Name	Kind	Name	Kind
CacqInt	Functional Block	MatxHWMsgMgr	Module
CtrackInt	Functional Block		
CxcorrInt	Functional Block		
CdspInt	Functional Block		
CatxReqInt	Functional Block	MatxHIMsgMgr	Module
CatxComInt	Functional Block		
Cinitialization	Functional Block	MatxMgr	Module
CdataStore	Functional Block		
CatxStrategy	Functional Block		
CatxHWAllocator	Functional Block		
PatxReq	Protocol	Iatx	Interface
PatxCom	Protocol		

[0208] Note that the ten component and protocol elements have been mapped to three module elements. Also note that the functional behavior for each module element is detailed by its encompassed component and protocol blocks.

- 5 [0209] Figure 25 shows a module architecture overview 2500 for the ATX Control module, including inter-task communication. The ATX Control module has been broken into the three modules 2502, 2504, and 2506 as detailed in Figure 25. Six interfaces 2508 (I-track), 2510 (I-acq), 2512 (I-xcorr), 2514 (I-dsp), 2516 (I-atx), and 2518 (I-timer) are shown. The notation of a solid line ending in a circle indicates a module which provides
- 10 an interface. Dotted lines with an arrow indicate the use of an interface by another module. Interfaces are identified by variables starting with a capital "I". In one implementation of Startup design, the GPS Rx Control Module controls the GPS sub-system resets. In this case, the ATX Control Manager via GPS Rx Control may call the

ATX Control Open and Close functions. In general, the ATX Control Interface (Iatx) module may be available to ATX Control Manager in the Applications layer.

[0210] The interfaces are explained below in Table 12:

Table 12
Iatx (2516) – This is the interface between the HI and HD sides of the software. ATX Control Manager may primarily employ the interface on the HI side.
Iacq (2510) – This interface may be offered by the Acquisition module to the rest of the ATX subsystem.
Itrack (2508) – This interface may be offered by the Track module to the rest of the ATX subsystem.
Ixcorr (2512) - This interface may be offered by the Xcorr module to the rest of the ATX subsystem.
Idsp (2514) – This interface may be offered by the DSP Manager module to the Platform layer, for example by the MatxHWMsgMgr module to determine memory allocation status.
Itimer (2518) – This interface may be offered by the HW Timer module to the Applications and Platform layers, for example, by the MatxHWMsgMgr, MatxHIMsgMgr, and MatxMgr modules for time tagging.

5

[0211] The HD to HI Interaction provides for the transfer of different types of data, including the following: Raw Measurement Data (including acq/tracking status, search strategy progress and recovery indicators); 50 BPS data (per satellite basis); WAAS data;

10 Acquisition Status Reporting.

[0212] Table 13 details the data block for the 50 BPS data transfer:

Table 13		
Field	Length (bytes)	Description
Sequence Number	1	For, e.g., logging and post-processing purposes.
Channel	1	Virtual Channel Assignment number.
PRN	1	PRN number of signal providing this demodulated data.
Data Collection Type	1	0x00 – subframe 0x01 – word Last four bits may be used to indicate which words are valid in the 50BPS data block (for 0x01 word case)
HW Time Tag	8	hardware time tag in AcqClk units
50BPS data[10]	4*10	10 words of data, each array element holds 30 bits (or 32 bits per word).

[0213] The “Channel” field may serve to indicate changes in the dedicated hardware
5 for that PRN. The “Data Collection Type” field may be employed for future expansion.
In one implementation, the architecture decodes a whole subframe after delivery,
although it may instead decode words in other implementations.

[0214] Table 14 details the data block for the WAAS data in the HD to HI side of the
ATX Interface.

Table 14		
Field	Length (bytes)	Description
Sequence Number	1	For, e.g., logging and post-processing purposes.
PRN	1	WAAS PRN number
newFrame	2	Flag to indicate good new frame
HW Time Tag	8	HW time tag in AcqClk units

Data[32]	1*32	Data bits, right justified in word
Report100ms	2	The 100ms report (0x800d048c for SSII)
SyncMeasure	2	
Resync	2	Number of resync since start
System 100ms	4	System time since power on.

[0215] Table 15 details the measurement block for the ATX Interface.

Table 15		
Sequence Number	1	For, e.g., logging and post-processing purposes.
Recovery flag	2	0x01 – Non-Visible PRN found 0x02 – Time error detected (<2s) 0x04 – Time error detected (>2s) 0x08 – Frequency Error detected 0x10 – Initial Acq incorrect, revert BEP 0x20 – Initial Bit sync incorrect, revert BEP 0x40 – Initial Frame sync incorrect, revert BEP
Reserved	1	For 4 byte alignment.
Raw Measurements * 16		
PRN	1	
Status	1	0x00 Idle 0x01 Acquisition Successful 0x02 Delta Carrier Phase Valid 0x04 Bit Sync Done 0x08 Subframe Sync Done 0x10 Carrier Pull-in Done 0x20 Code Locked 0x40 Acquisition Failed
Extended Status	1	0x01 Is this a Recovery PRN (non visible) 0x02 First acquisition 0x04 First bit sync 0x08 First Frame Sync 0x10 Xcorr eliminated flag 0x20 Multipath reduction flag
Acq Parameters for current search	4	This will be a duplicate of information available in status block for this SV. MF Locked Mode:1 0b Unlocked (1 msec T1) 1b Locked (1/4 msec T1)

		MF Fractional Mode:2 00b Full (1 chip spacing) 01b Frac (½ chip spacing) 10b Frac (¼ chip spacing) 11b Frac (1/8 chip spacing) MF High Resolution:1 0b Low resolution 1b High resolution MF Half Chip Enable:1 0b Disabled 1b Enable (requires second pass) FFT Mode:2 00b 8 pt 01b 16 pt 10b 32 pt PDI:5 Actual PDI value in msec pad:4 NCS:16 Actual NCS period in msec
Error report		
Ms ambiguity range		Range of potential ms error, limited to 100ms.
Meas consistency		Relative PR may be consistent.
HW Time Tag	8	In AcqClk units
Msec number	2	Number of msec into current bit. units: 1 ms Range: 0 to 19
Bit number	4	Number of downlink data bits into the current GPS week. units: 20 ms Range: 0 to 30,240,000
Code Offset	4	Represents time in units of L1 carrier cycles units: 1/1024 L1 cycles Range: 0 to 1540*1023*2**10 L1 cycles
Carrier Phase	4	Represents time in units of L1 carrier cycles units: 1/1024 L1 cycles Range: 0 to 1540*1023*2**10 L1 cycles
Carrier Doppler	4	Doppler frequency units: 1 / 2**21 cycles/msec Range: 0 – (1 / 2**32 – 1) cycles/msec Note: This is 1/2097.152 cylces/sec
Carrier Doppler Rate	4	Doppler Frequency Rate units: 1 cycle / iter**2 * 2**5

C/No	1	100 ms estimate
Delta Range Interval	1	Number of ms in the Carrier Doppler measurement
Phase Error Count	1	Number of 20 ms intervals with carrier phase error > threshold
Low Power Count	1	Number of 20 ms intervals with C/N0 < 31 dBHz

[0216] Table 16 details the conditioned measurement block for the ATX Interface.

Table 16		
Sequence Number	1	For, e.g., logging and post-processing purposes.
Recovery flag	2	0x01 – Non-Visible PRN found 0x02 – Time error detected (<2s) 0x04 – Time error detected (>2s) 0x08 – Frequency Error detected 0x10 – Initial Acq incorrect, revert BEP 0x20 – Initial Bit sync incorrect, revert BEP 0x40 – Initial Frame sync incorrect, revert BEP
Reserved	1	For 4 byte alignment.
Conditioned Measurements * 16		
PRN	1	
Status		0x00 Idle 0x01 Acquisition Successful 0x02 DeltaPhase Valid 0x04 Bit Sync Done 0x08 Subframe Sync Done 0x10 Carrier Pull-in Done 0x20 Code Locked 0x40 Acquisition Failed
Extended Status		Is this a Recovery PRN (non visible) First acquisition First bit sync First Frame Sync Xcorr eliminated flag Multipath reduction flag
Acq Parameters for current search		This may be a duplicate of information available in status block for this SV. MF Mode FFT Mode PDI NCS

Error report		
Ms ambiguity range		Range of potential ms error, e.g., up to 100ms.
Meas consistency		Relative PR may be consistent.
HW Time Tag	8	In AcqClk units
PR		
Carrier Freq		
Carrier Phase		
Time in Track		
Sync Flags		
C/No 1		
C/No 2		
C/No 3		
C/No 4		
C/No 5		
C/No 6		
C/No 7		
C/No 8		
C/No 9		
C/No 10		
Delta Range Interval		
Mean Delta Range Time		
Extrapolation count		
Phase Error Count		
Low Power Count		

[0217] Table 17 details the acquisition status block for the ATX interface that contains information on the progress of the search strategy. For each PRN there may be multiple acquisition reports. The acquisition reports may be generated when the GPS Rx Control requests them. The acquisition reports may be used by the power management logic and the CP-Centric code, as examples. In one implementation will output acquisition reports each time a search is started that may be queued by the ATX Control block. The GPS Rx Control module may then check for acquisition reports, for example, every 100ms (for system information). If acquisition reports are available, the GPS Rx

Control may then retrieve the queued acquisition reports. The GPS Rx Control module may store (and provide an interface to retrieve) the acquisition data.

Table 17		
Field	Length (bytes)	Description
Sequence Number	1	For, e.g., logging and post-processing purposes.
PRN	1	WAAS PRN number
Acquisition Type		
Acq Parameters for search in Progress		Same as field in measurement block MF Mode FFT mode PDI NCS
Expected time to completion		0-15 seconds (maybe longer?)
Timetag		Ms number
Code uncertainty in Search		0-1023 chips
Frequency Uncertainty in Search		0-12 ppm
C/No List in Search		C/No list
Code Range Searched		0-1023 chips
Freq Range Searched		0-12 ppm
Time to complete full search for all PRNs		Time to complete the full search strategy for all PRNs.

5

[0218] The discussion below explains the HI to HD direction of the ATX Interface.

The HI side of the software may deliver the following data structure to the HD side of the

10 software. In one implementation the ATX interface may operate using a block copy.

The ATX interface may then employ the "New Data" field to determine which fields to copy over (assuming the "Visible List" field has not changed). The "Event" field may be synchronized to selected Rx Controller events. Table 18 explains the HI to HD messaging.

5

Table 18		
Field	Length (bytes)	Description
Sequence Number	1	For, e.g., logging and post-processing purposes.
Mode Command	1	Used to command indicate what type of mode to use. 0x00 – default 0x01 – common uncertainty 0x02 - testmode The common mode may be employed when every PRN has the same uncertainty. When this mode is indicated, the ATX block obtains the visible List info from the visible List field and then obtains the uncertainties and search centers from the first prepositioning entry. Testmode may tell ATX block to use first Prepositioning Entry for testing the HW.
Memory Mode Request	1	This field may give the ATX block instructions on how to prioritize memory use between the acquisition, tracking and xcorrelation detection blocks. This block may give basic high level information, with the ATX block controller specifying details. Alternatively, this field may be employed to identify customer memory availability.
Measurement Rate Request	1	May be employed to reduce throughput, e.g., for serial processing in Tracker.
WAAS Prn number(s)	1	Field representing current WAAS prns.
Event	3	Event field, multiple events are possible. 0x0001 – All PRN records have same time stamp 0x0002 – Initial Command 0x0004 – Periodic Update 0x0008 – First Navigation 0x0010 – External Aiding Update 0x0020 – Artificial Bit Sync

		0x0040 – Artificial Frame Sync 0x0080 – Internal Aiding Update 0x0100 – First Acquisition Considered 0x0200 – First Acq Used 0x0400 – Bit Sync Considered 0x0800 – Bit Sync Used 0x1000 – Frame Sync Considered 0x2000 – Frame Sync Used 0x4000 – New Nav Edit 0x8000 – Forced Recovery
Visible List	4	Bit field of visible satellites. May be employed in conjunction with “common” type in the mode command field. Bit number is SV-1.
Recovery List	4	Bit field of satellites on the recovery list. Indicates all remaining healthy satellites outside of the visible list above. Bit number is SV-1.
Channel Edit	4	Bit field of satellites indicated by Nav to have errors. ATX block may attempt to find better signal and ignore the current tracked signal. Bit number is SV-1.
New Data	4	Bit field of satellites which have new data in this message. Employed when updating the ATX block with new information. For instance, receiving ephemeris for a given satellite. That prepositioning information may be updated in the message and the corresponding bit set to 1 in this field. Any bit here may also be set in the Visible List field. Bit number is SV-1.
Augment list	4	This is a space holder for future expansion that may be employed when tracking signal sources with PRN numbers outside of the 1-32 range. Some mapping may be employed (e.g., the first four bits representing pseudolites, starting at 36). Bit number is SV-1.
Suspected Multipath SV List	4	Bit field indicating GPS PRN numbers which have potential multipath as determined by the HI layer. Bit number is SV-1.
User Time	8	Double Value representing the current user time. May be employed by ATX block for recovery situation.
HW Time Tag	4	The value of the HW timer corresponding to the User TOW. Assumes that RTC interface on SiRFNav can provide RTC registers and

		corresponding HW time tag.
GPS Week	2	Extended GPS Week number. May be employed by ATX block for recovery situation.
Search Status Request	1	Requests that the ATX block start sending search status messages on the ATX Interface.
Reserved	1	For 4 byte alignment. In future this field may be used to indicate how many PRN records are included (for serial transmission case, for example).
Velocity	2	Estimated user velocity units: m / sec
Velocity Uncertainty	2	User velocity uncertainty (one sided) units: m / sec
Clock Drift	2	Estimate clock drift units: Hz
Clock Drift Uncertainty	2	Clock drift uncertainty (one sided) units: Hz
Satellite Record*16		
PRN	1	PRN number for current satellite record.
Bit Fields	1	0x01 – multipath detected 0x02 – xcor detected
C/No 1 Value	1	First C/No search level. [db/Hz]
C/No 1 Prob	1	Probability that signal is above C/No 1. [%]
C/No 2 Value	1	Second C/No search level. [db/Hz]
C/No 2 Prob	1	Probability that signal is above C/No 2. [%]
C/No 3 Value	1	Third C/No search level. [db/Hz]
C/No 3 Prob	1	Probability that signal is above C/No 3. [%]
HW timetag	8	HW timer ms number associated with record.
PR	8	Expected pseudorange, double value. [m]
PR Unc	4	Pseudorange expected uncertainty, float value [m].
PR Prob 1	1	Probability PR is within expected uncertainty [%]
PR Prob 2	1	Probability PR is within twice the expected uncertainty. [%]
PRR	4	Expected Pseudorange rate, float value [m/s].
PRR Unc	4	PRR expected uncertainty, float value [m/s].
PRR Prob 1	1	Probability PRR is within expected uncertainty [%]
PRR Prob 2	1	Probability PRR is within twice the expected uncertainty. [%]
Sub1ms Unc	2	0-512 [chips]
Sub1ms Prob 1	1	Probability sub1ms is within expected uncertainty [%]
Sub1ms Prob 2	1	Probability sub1ms is within twice the expected uncertainty. [%]
Reserved	1	For byte alignment.

End of PRN Records		
Cross-Correlation Table	32	
Nav Bit Aiding		

[0219] Table 19 shows the C/No for each PRN and reflects a uniform distribution as follows:

5

Table 19		
#	C/No Value	C/No Probability
1	36	33
2	24	66
3	12	100

[0220] The probabilities for PR, PRR and Sub 1ms may be set to reflect that the value is assumed to be 100% within the uncertainty. This means the "Prob 1" and "Prob 2" fields may all be 100% in certain implementations. The entries in the event field are described below. Note that some bit entries may remain active after the event has occurred in order that the ATX Control module does not miss certain events if a record is missed. If the ATX Control module has not used the previous prepositioning block, the new block may indicate all events in the previous and current prepositioning blocks (i.e. the ATX Control module may get at least one block indicating events which have occurred since its last update of the prepositioning block).

15

[0221] All PRN records may have a common timestamp in order to indicate to the ATX Control module that all the PRN records have been updated to the same time stamp. The ATX Control Module may employ prepositioning data with a common time stamp to generate information on inter-measurement relationships. This bit may be set when the

Initial command is sent, for periodic updates, for first navigation and potentially for internal or external aiding.

[0222] Initial Command: The initial command may be the first block of data sent to the ATX Control module. The ATX Control module may employ this as a trigger for a selected initialization. The “All PRNs updated” flag may be set when the initial command bit is set. This bit may be turned off after the initial command has been sent.

[0223] Periodic Update: The Rx Controller may send new prepositioning information to the ATX Control module every 5 seconds, for example. This updated data may reflect propagation of the system level uncertainties. If a periodic update is performed, the “All PRNs updated” flag may be set as well. This bit may be set for the periodic update, but not left active.

[0224] First Navigation: The Rx Controller may update the prepositioning uncertainties after navigation has occurred and the GPS Rx Control module has updated the internal uncertainties with the navigation information. Since navigation may affect all prepositioning data, the “All PRNs updated” flag may be set at the same time. Once first navigation has occurred, this bit may remain set.

[0225] External Aiding Update: This bit may be set if external aiding has been used to improve the uncertainties in the system. If the external aiding affects all satellite records in the prepositioning block, the “All PRNs updated” bit may be set as well.

[0226] Artificial Bit sync: This bit may be set if a time transfer received from external aiding or the internal system time allows for an artificial bit sync (e.g., if system time uncertainty is less than 250 us, for example). If this bit is set, the external aiding

update bit may be set as well. The "All PRNs updated" may be set if the external time transfer improved all prepositioning information. This bit may remain set (unless there is a recovery situation indicated).

[0227] Artificial Frame Sync: This bit may be set if an external time transfer or the
5 internal system time allows for artificial frame sync (e.g., the system time is less than 5 ms). If this bit is set, the external aiding bit may be set as well. This bit may remain set (unless there is a recovery situation).

[0228] Internal Aiding Update: This bit may be set if there was an internal aiding
10 update. If the internal aiding update affected all satellites then the "All PRNs updated bit may be set as well.

[0229] First Acquisition Considered: This bit is part of the handshaking between the
ATX Task and the Rx Controller Task. The ATX Task employs a mechanism to revert to
an earlier set of system level uncertainties if an acquisition, bit sync or frame sync proves
incorrect. The ATX task will indicate to the GPS Rx Control module that a first
15 acquisition has occurred using the HD to HI ATX Interface. The GPS Rx Control may
then pass the aiding information to the BEP module. The BEP module may then be
responsible for maintaining up to three different values for the freq and time uncertainty.
The Rx Controller Task may set this bit if the first acquisition data from the tracker was
passed to the aiding module (not necessarily used to set system uncertainties). Thus, the
20 ATX Control module may know that the acquisition data was considered for the current
set of prepositioning data. This bit may remain set unless the ATX Control module
indicates that the initial acquisition was incorrect (using recovery field in the HD to HI

ATX Interface). If the initial acquisition is incorrect, the GPS Rx Control may request that the BEP module revert to the pre-acquisition system uncertainties for time and frequency.

5 [0230] First Acq Used: This bit may be used in a manner similar to the "First acq considered" bit except it may be set if the acquisition data from the ATX Control module was actually used to improve the system uncertainties.

[0231] Bit Sync Considered: May be employed in a manner similar to the "First Acq considered" bit but for bit sync.

10 [0232] Bit Sync Used: May be employed in a manner similar to the "First Acq Used" bit but for bit sync.

[0233] Frame Sync Considered: May be employed in a manner similar to the "First Acq Considered" bit but for Frame sync.

[0234] Frame Sync Used: May be employed in a manner similar to the "First Acq Used" bit but for Frame sync.

15 [0235] Nav Edit: This bit may be set if the HI side of the SW determines that a measurement is invalid. It may be used in conjunction with the channel edit list to indicate the problem PRN number. This bit may be reset after each message.

[0236] Forced Recovery: This bit may be set if the prepositioning data has been updated due to a recovery situation.

20 [0237] The ATX Interface may provide a function (or sequence of functions) that provide estimates of time to complete a search strategy given a certain level of system uncertainty. These functions may be called from the Serial Input Task Thread when CP

centric message requests are received. The CP centric function handlers may be in the SL module. When an estimated time to complete a search strategy is employed, the function will take as arguments the following: Number of PRNs, Number of PRNs with ephemeris, prepositioning uncertainties for non-ephemeris case, Prepositioning
5 uncertainties for ephemeris case, and Desired Sensitivity.

[0238] The requesting module may generate the satellite prepositioning uncertainties based on the current system level uncertainties or a proposed set of system level uncertainties. The BEP module may provide the extrapolation and prepositioning functionality for the requesting module. The sequence may follow the steps shown in
10 Table 20.

Table 20	
1)	Requesting module retrieves system level uncertainties from the BEP module.
2)	Requesting module adjusts system level uncertainties to generate a hypothetical set of uncertainties if desired.
3)	Requesting module uses BEP module to generate prepositioning data for the ephemeris and non-ephemeris case.
4)	Requesting module uses ATX Interface to request the max search time for the given uncertainties, number of PRNs and required sensitivity.
5)	ATX module returns max search time.

[0239] The ATX interface function may return the maximum search time to complete the full search strategy (all PRNs). The maximum search time may be calculated

assuming that no searches are used which have higher sensitivity than the indicated “required sensitivity”.

[0240] The purpose of retrieving the maximum search time is to determine whether injecting better information into the system may improve the time-to-first-fix (TTFF). If the system cannot reach the requested sensitivity based on the proposed uncertainties, it may indicate that fact in a return value. If the resultant maximum search time based on the hypothetical system uncertainties is acceptable, the requesting module may then inject the appropriate information into the system using the normal aiding procedure (e.g., requested across a network for example) and then employ the HI to HD ATX Interface C/No Table to limit the search sensitivity to the requested sensitivity.

[0241] The Iatx interface may be implemented in the MatxHIMsgMgr module which includes the blocks related to the PatxReq and PatxCom protocols. The “ModuleIdEnum” in the descriptions represents the module ID of the calling module. The following structure in Table 21 may be defined in the UI Debug module to allow logging of the inter-module interfaces. It may be passed as a pointer to allow for easy modification of the structure type and to ensure the stack requirement for the data is only the pointer.

Table 21
<pre>Typedef struct { UIDbg_moduleIdEnum id; /* will be one byte long */ INT reserved[3]; } UIDbg_ChnlInfoType;</pre>



[0242] Exemplary Iatx Interface functions are described below in Table 22:

Table 22
WERR atxControl_Close(UIDbg_ChnlInfoType *pSrc)
WERR atxControl_Open(UIDbg_ChnlInfoType *pSrc, moduleCommandEnum comType)
WERR atxControl_Command(UIDbg_ChnlInfoType *pSrc, int *pSeqNum)
WERR atxControl_Request(UIDbg_ChnlInfoType *pSrc, int *pSeqNum)

5

[0243] Close: Sets the ATX Control module status to closed. Returns FAILURE if the ATX Control module is currently updating at a lower task priority. As part of closing the ATX Control module, it will call the close functions for the Acquisition, Track, Xcorr, and DSP Manager modules. This may be called at the lowest priority.

10 [0244] Open: This function will attempt to Open and initialize the ATX Control module. Returns FAILURE if it cannot open the module or if the module is already open. As part of opening the ATX Control module, it will call the open functions for the Acquisition, Track, Xcorr, and DSP Manager modules. This open function may initialize the ATX Control module, but will not start any acquisitions. A separate command
15 function may be invoked to start acquisitions.

[0245] The comType argument may be provided for modules which may have a local non-volatile store. It may take two values (e.g., MC_DEFAULT, MC_LATEST). The MC_DEFAULT command will force the module to clear its local memory. A value of MC_LATEST may be treated as MC_DEFAULT.

5 [0246] Command: This is the command function for the ATX Control module. It may be used to inform the ATX Control module that new or updated commands are available.

[0247] Request: This is the request function for the ATX Control module. It may be used by the ATX Control Manager to request acquisition status from the ATX Control
10 module.

[0248] The control of the ATX Control module may be handled through the ATX task 2600 as shown in Figure 26, including the steps 2602 and 2604. The ATX task 2600 may be scheduled from a selected interrupt service routine (ISR).

[0249] Figure 27 shows a flow diagram 2700 of the execution behavior of the ATX
15 Control module. Figure 27 shows that the high level function of the ATX Control module can be divided into two parts: controlling the interfaces 2702 – includes the Iatx interface and the interfaces with the Acquisition, Track, Xcorr and DSP Manager modules; determining the search strategies 2704 – for the commanded acquisitions, use the status information from the Acquisition, Track, Xcorr and DSP Manager modules to
20 determine the best method to provide measurements for navigation.

[0250] The interface manager control 2702 includes: Process status from Acquisition, Track and Xcorr modules (2706), Check for request for acquisition status and create

message if desired (2708); Create output messages (2710); Schedule Rx Controller task when measurements available or 100 ms boundary (2712); Check Iatx interface for new or updated SV commands (2714); process records (2716).

[0251] Figures 28, 29, and 30 show flowcharts 2800, 2900, and 3000 of the interface manager processing. Status from current and completed satellite commands are
5 processed next. This information is employed for several tasks – acquisition status request, output messages, and determining search strategy. The status is collected from the Acquisition, Track and Xcorr modules.

[0252] The Acquisition Status Request is a message employed by the Power Manager
10 Logic module. The message includes an estimate of the time to complete all satellite commands.

[0253] The Output Messages include: Measurement Data, 50 bps satellite Data and WAAS Data. The Measurement Data message may be output at 1 Hz, for example. The 50 bps SV Data may be output once per subframe per satellite (6 seconds), up to once per
15 word per satellite (0.6 seconds). The WAAS Data may be output once per second when a WAAS PRN is being tracked.

[0254] The Rx Controller Task Scheduling: The Rx Controller Task may be asynchronous and may be scheduled under selected circumstances. For example, when measurements are available or when a 100 ms boundary occurs.

20 [0255] The ATX Control Manager Module may place new commands and update old commands in the Iatx interface. Bit fields in the satellite command message indicate new or updated commands. Updated commands may be frequent due to improved

prepositioning estimates due to, but not limited to, successful acquisitions, external aiding or a converged navigation solution.

[0256] The ATX Control Module may scan the message for new and updated commands. This information may be copied to the ATX Control Module local memory
5 to be employed by the search strategy manager.

[0257] Extensive capabilities are provided to acquire satellites.

[0258] Two modes of operation for the search strategy manager are the acquisition mode and the track/acquisition mode. On startup and reset, the default mode is acquisition. Once a satellite has been acquired and tracked, the mode will switch to
10 track/acquisition mode. It will remain here until the ATX Control Module is closed.

[0259] Figure 31 shows a high level state machine 3100 including states 3102-3110 for the search strategy manager, described below. Transitions may be made on events (as opposed to time, for example).

[0260] The System_Idle State (3102): This is the default state after startup or reset.
15 When in this state, the ATX Control module is waiting for an open request. When it is received, control is transitioned to the System_Open state.

[0261] The System_Open State (3104): When in this state, the ATX Control module is waiting for the first set of SV records. When they are received, control is transitioned to the System_Acq state.

20 [0262] The System_Acq State (3106): When in the System_Acq state, the satellite command message is processed to determine which satellites and how many satellites to command at a time. Then, the commands are sent to the Acquisition module and ATX

Control module waits for status. When an acquisition fails, the satellite is recommended with a longer dwell. On the first successful acquisition, the mode state variable is transitioned to the System_Track state.

[0263] The System_Track State (3108): While in the System_Track state, tracking
5 satellites have priority over acquisition satellites. After the in track satellites are processed, new track satellites are processed and allocated HW resources. Then, the satellite acquisition commands are processed and commanded using the remaining DSP RAM.

[0264] The System_Close State (3110): When in any state, a close request will cause
10 the state to transition to System_Close. All active acquisitions and tracks will be closed and the state transitioned to System_Idle.

[0265] One primary purpose of the Rx control module is to control the data updates and execution of the Rx Control task. The discussion below explains the how the control module controls data updates and execution of the Rx control task to service interaction
15 between hardware dependent (HD) and hardware independent (HI) components of the overall software in the architecture, for example, to maintain measurement and acquisition search information.

[0266] The control module may also handle recovery situations when indicated by the ATX block and perform internal aiding based on data received from ATX block. The
20 control module may update the BEP, satellite Data and Visible List modules as well as perform internal aiding based on NAV data in the previous Rx Control task execution.

[0267] Abbreviations employed below may include: HD - Hardware Dependent; HI – Hardware Independent; QoS – Quality of Service; Rx – Receiver; and SSB- Software Binary.

[0268] Table 23 shows optional features of the control module, in one
5 implementation.

Table 23
Feature
GPS Rx Control may call the open function of the GPS sub-system at startup.
GPS Rx Control may generally run periodically at 100 ms but may be scheduled to run asynchronously.
GPS Rx Control may schedule the Back Ground Task every 1000 ms.
GPS Rx Control may be responsible for maintaining measurement information from ATX.
GPS Rx Control may maintain integrity of the data passed to ATX.
GPS Rx Control may call ATX Ctrl Mgr to generate commands for ATX Control.
GPS Rx Control may call Navigation and Aiding for Quality of Service calculations.
GPS Rx Control may call Power Manager Logic.
GPS Rx Control may service error conditions from UI Debug.
GPS Rx Control may handle recovery information from the ATX.

10 [0269] Figure 32 shows a relation diagram 3200 for relations between the GPS Rx Control module and the rest of the architecture modules 3202. The relation diagram 3200

includes interaction between the module of interest and other modules (actors). Interactions may be of several types including: uses (calls), creates, updates and obeys. The use case diagram 3200 identifies modules and runtime elements.

[0270] The relationships that exist in Figure 32 are summarized below in Table 24.

5

Table 24
Startup: Commands the module to initialize (Open command).
Reset: Commands the module to reset, generally as part of a GPS Start/Stop sequence.
Background Task: GPS Rx Control schedules for periodic update.
UI GPS: GPS Rx Control module may signal an event to the UI GPS module when the GPS Rx Control module updates internal information or navigation has completed (nominally 1 second). The event may cause the UI GPS module to output pre-selected binary messages.
MI GPS: May request measurement and acquisition state information from GPS Rx Control module for output.
Aiding: When navigation has completed, GPS Rx Control module may call aiding for internal updates. Aiding may then update BEP, satellite Data and Visible List.
ATX Ctrl Mgr: GPS Rx Control module will receive HD->HI information from the ATX output and send HI->HD information to the ATX input.
BEP: GPS Rx Control module commands force update when significant new aiding data is received.
Ctrl App: GPS Rx Control may get mode and control information from the Ctrl App module.
DGPS App: GPS Rx Control may send SBAS data to the DGPS App module.
NAV: GPS Rx Control module may command NAV to update with new measurement inputs.

Power Mgr Logic: GPS Rx Control may call Power Mgr Logic for Trickle Power management and APM type power management.

QoS: When navigation has completed, GPS Rx Control may call aiding for QoS processing.

SV Data: GPS Rx Control module commands force update when significant new aiding is received.

Visible List: GPS Rx Control module commands force update when significant new aiding is received.

[0271] The GPS Rx Control module may support the following types of queries and commands as part of the use case scenarios shown in Table 25.

Table 25	
Command/Query	Description
Open request.	Request to open the module, may also force initialization of module. GPS Rx Control may also open the other modules of the GPS sub system.
Close request.	Request to close the module. GPS Rx Control module may be responsible for closing the other modules of the GPS sub system.
Command Aiding to run QoS function.	After navigation, command Aiding to run the QoS function with the navigation results.
Command Aiding to update internal aiding.	Run Aiding to update internal aiding with the navigation results.
Command ATX Ctrl Mgr to create ATX Input.	Command to fill ATX Block including Pre position data.

Command BEP update.	Force update of BEP data.
Command Navigation to run NAV update.	Call Navigation to update navigation solution with the latest measurements.
Command SV Data update.	Force update of SV Data.
Command UI Output of Navigation solution.	Upon completion on navigation, Rx Control may set the Event Flag for Navigation output.
Command Visible List update.	Force update of Visible List.
Query ATX Ctrl Mgr for measurement data.	Retrieve measurement data.
Query UI Debug for error conditions.	Request to service the error conditions.
Schedule Back Ground Task.	When appropriate, Rx Control may schedule BG every 1000 ms.
Schedule Rx Control Task.	At the end of a process cycle, if Events for Rx Control are outstanding, Rx Control may schedule itself to run again.
Transfer measurement data to Aiding.	Rx Control may transfer 50 BPS data to Aiding.
Transfer DGPS data to DGPS App.	Rx Control may transfer SBAS data to DGPS App.

[0272] Table 26 shows module specific implementation options.

5

Table 26	
Feature	Implementation
Make initialization explicit to ensure every module has the capability to initialize itself	Thus, the architecture may provide an initialization. (create CInitialization

on demand	block).
Isolate the Core functionality in a module from its interfaces	Thus, the architecture may isolate core update code from the external protocols by providing interface blocks (create interface blocks CrxControlInputHandler, CrxControlOutputHandler, CatxHandler, CaidingHandler, CbepHandler, CctrlAppHandler, CdgpsAppHandler, CnavHandler, CpowerMgrHandler, CqoSHandler, CsvDataHandler, CuiHandler, CvisListHandler, CmsgHandler, CioHandler) and (create core block CdataStore, CctrlUpdate blocks)

[0273] Table 27 shows an exemplary set of protocols that the control module may interact with.

Table 27
<p>PrxControlInput –</p> <p>PrxControlOutput –</p> <p>PAtxCtrl – Interface to the ATX Ctrl Mgr module.</p> <p>Paidding – Interface to the Aiding module.</p> <p>Pbep – Interface to the Best Estimator and Pre Positioning module.</p> <p>PctrlApp – Interface to the Ctrl App module.</p> <p>PdgpsApp – Interface to the DGPS App module.</p> <p>Pnav – Interface to the NAV module.</p> <p>PpowerMgr – Interface to the Power Mgr Logic module.</p> <p>PqoS – Interface to the QoS module.</p> <p>PsvData – Interface to the SV Database block.</p>

Pui – Interface to the User Interface module for outputting measurements, search status.

PvisList – Interface to the Visible List module and defined in the Visible List Module.

[0274] . Note that a protocol may be specified (including handshaking) from the point
5 of view of a particular functional block. Other functional blocks which wish to
communicate may use an opposite or conjugate protocol indicated by the suffix “-conj”.
Components (functional blocks) in the conceptual view are denoted with a starting capital
“C”.

[0275] Figure 33 shows a block diagram of the control module 3300, including a
10 message handler component 3302, a control input handler 3304, a satellite data handler
3306, as well as the other components illustrated. Figure 34 shows a block diagram 3400
of a message handler 3302 portion of the control module. The message handler 3302 may
include an I/O handler component 3402, an initialization component 3404, a data store
component 3406, and a control update component 3408.

15 [0276] A protocol description for PrxControl is provided in Table 28, while a
protocol description for PrxControlOutput is provided in Table 29.

Table 28	
<<Protocol>>	
PrxControlInput	
Incoming	
TBD	
Outgoing	
....	
Sequencing (CrxControlInputHandler into ←, out of →)	
Direction	
←	TBD

Table 29	
<<Protocol>>	
PrxControlOutput	
Incoming	
....	
Outgoing	
Rx Control Output (Internal aiding, Measurements, 50 BPS, SBAS data, Acquisition Status, ATX status)	
Sequencing (CrxControlOutputHandler, into ←, out of →)	
Direction	
→	Rx Control Output

5

[0277] The module architecture from a layer point of view may be of low complexity
10 because the GPS Rx Control module may be implemented to be essentially hardware independent. The interaction between the GPS Rx Control module and its actors may

also be done at similar tasking levels so no breakdown for fast versus slow processing becomes necessary. Table 30 shows a mapping for module elements.

Table 30			
Conceptual Elements		Module Elements	
Name	Kind	Name	Kind
CrxControlInputHandler	Functional Block	MmsgMgr	Module
CrxControlOutputHandler	Functional Block		
CaidingHandler	Functional Block		
CatxMgrHandler	Functional Block		
CbepHandler	Functional Block		
CctrlAppHandler	Functional Block		
CdgpsAppHandler	Functional Block		
CioHandler	Functional Block		
CnavHandler	Functional Block		
CpowerMgrHandler	Functional Block		
CsvDataHandler	Functional Block		
CuiHandler	Functional Block		
CvisListHandler	Functional Block		
Cinitialization	Functional Block	MrxControlMgr	Module
CdataStore	Functional Block		
CctrlUpdate	Functional Block		
PrxControlInput	Protocol	Irx	Interface
PrxControlOutput	Protocol		

5

[0278] Figure 35 details the module architecture view 3500 for the GPS Rx Control module, showing modules (e.g., the QoS module 3502 and the Navigation module 3504) that may be included in the architecture.

10 [0279] The Rx Control module is the execution loop of the GPS sub-system 3506. It employs the interface of all of the modules in the GPS sub-system 3506 and also many of the modules in the Application layer 3508. Interfaces are identified by variables starting

with a capital "I" (e.g., the receive control interface Irxc 3512). Rx Control may control the GPS sub-system modules' resets. It may also be responsible for calling the Close and Open functions in a pre-determined sequence. The Rx Control interface may be available in the Application layer, while the ATX Ctrl Mgr will transfer data to and from the Platform layer 3510.

[0280] Note that the IrxControl interface is implemented in the MmsgMgr module and that the MmsgMgr module includes blocks related to PrxControlInput and PrxControlOutput protocols. The Irxc interface 3512 may be specified as shown below in Table 31.

Table 31
<pre>/* Reset Rx Control and GPS Sub system. Where mode: 0 = Full, 1 = Partial. */ WERR rxc_Reset(UIDbg_ChnlInfoType *pSrc, short mode); Run rxc_Close() and rxc_Open(). /* Close Rx Control and close GPS sub system. Where mode: 0 = Full, 1 = Partial. */ WERR rxc_Close(UIDbg_ChnlInfoType *pSrc, short mode); /* Open Rx Control and open GPS sub system. */ WERR rxc_Open(UIDbg_ChnlInfoType *pSrc);</pre>

[0281] Rx Control may close the following modules: ATX Ctrl, MI GPS, Navigation, DGPS Apps, Visible List, satellite Data and BEP. For a partial close, BEP, satellite Data

and DGPS App may not be closed so they can remain open and receive updates from external sources. Rx Control may open the following modules: ATX Ctrl, MI GPS, Navigation, DGPS Apps., Visible List, satellite Data and BEP.

[0282] Rx Control may store the ATX satellite Data and SBAS raw data for use by
5 the various Hardware Independent modules. ATX Ctrl Mgr may copy the raw data from the Iatx interface to a matching structure in Irxc.

[0283] Rx Control may also store the NAV measurement data. The NAV measurement data is then filled by ATX Ctrl Mgr, which converts the raw ATX Ctrl measurements to the NAV measurement format. The NAV measurement structure may
10 also be output on a serial port as a pre-defined message.

[0284] Table 32 shows prototypes for a measurement obtaining call and a acquisition status obtaining call.

Table 32
<pre>/* Returns NAV formatted measurement structure */ WERR rxc_GetMeasurements(UIDbg_ChnlInfoType *pSrc, tNavMeas *); /* Returns ATX Ctrl structure for Acquisition Status */ WERR rxc_GetAcquistionStatus(UIDbg_ChnlInfoType *pSrc, tAcquistionStatus *);</pre>

15 [0285] The NAV measurement structure shown in Table 33 may be employed in the architecture. This structure is dimensioned [NUM_OF_MEASUREMENTS].

Table 33
NAV Measurement Structure

```
{
    short                // Measurement Sequence Number;

    UINT32 Timetag;      // ACQCLK lsw
    UINT32 Timetag2;     // ACQClk msw

    double measTOW;      // User time

    UBYTE SVID;          // Sat ID for each channel

    double Pseudorange;  // Pseudorange in meters

    float  CarrierFreq;  // Pseudorange rate in meters/seconds

    double CarrierPhase; // Integrated carrier phase in meters.

    short  TimeInTrack;  // Count, in milliseconds, of how long sat is in track

    UBYTE SyncFlags;     // This byte contains two bit-fields which report the
                        // integration interval and sync achieved for the channel.
                        // Bit 0: Coherent Integration Interval (0=2ms, 1 = 10ms)
                        // Bits 1,2: Sync

    UBYTE CtoN[10];      // Average signal power in db-Hz for each 100 ms

    UINT16 DeltaRangeInterval; // Interval for the preceding second. A value of zero
                        // indicates that we have an AFC measurement or no measurement in the
                        // CarrierFreq field for this channel.

    INT16 MeanDeltaRangeTime; // Mean time of the delta-pseudo range interval in
                        // milliseconds measured from the end of the interval backwards.

    INT16 ExtrapolationTime; // The pseudo range extrapolation time in
                        // milliseconds, to reach the common Timetag value.

    UBYTE PhaseErrorCount; // This is the count of phase errors greater than 60
                        // deg measured in the preceding second (as defined for each channel).

    UBYTE LowPowerCount;  // This is the count of power measurements less than
                        // 28 dB-Hz in the preceding second (as defined for each channel).

#ifdef FALSE_LOC
    double TrueRange;    /* true range */
}
```

```
long GPSSecond;          /* Integer GPS seconds */

long ClockOffset;        /* clock offset in Hz */
#endif

char  MeasurementConsistency;  Flag to indicate measurements are consistent
double Validity Time;         Receiver Time of Validity
short PRQuality;              Same definition as SiRFStar 2
float PR noise;               1 sigma expected PR noise in meters
float PRR noise;              1 sigma expected PRR noise in meters/sec
short PRRQuality;             Same definition as SiRFStar 2
float Carrier Phase Noise;    1 sigma expected Car Phase noise in meters

short PowerLockCount;         Count of Power Lock Loss in 1 second 0-50
short CarrierLockCount;       Count of Phase Lock Loss in 1 second 0-50
short msAmbiguity;            Millisecond ambiguity on measurement

//{
//    extension for DGPS, WAAS
//}

} tNavMeas;
```

[0286] Figure 36 shows a flow diagram 3600 of initialization steps reset 3602, initialize 3604, open 3606, schedule 3608, and completed 3610 for the control module.

5 [0287] Figure 37 shows a flow diagram 3700 that details the open procedure open steps 3702-3718 for the GPS Rx Control Module. The close procedure follows a similar execution flow in the reverse order.

[0288] Figure 38 shows a flow diagram 3800 of GPS Rx Control Module Execution, including the steps 3802 - 3820.

10 [0289] Figures 39-41 shows interaction diagrams 3900, 4000, and 4100 that illustrate the interaction (e.g., the interaction labeled 3912, as examples, message passing, interrupt

servicing, and the like, between the various functional blocks 3902-3910 in the architecture. The functional blocks include a message handler 3902, a command interpreter 3904, a data storage control block 3906, an update control block 3908, and an output message handler 3910.

5 **[0290]** A basic Expert System is a system that may be implemented as a state machine or control process that receives data about the operation of the system and takes actions according to the data. The Expert System may be adaptive in nature and change how it functions over time. In a satellite positioning receiver, an Expert System may coordinate the control and operation of the satellite positioning receiver.

10 **[0291]** In figure 42, a satellite positioning system receiver 4200 is shown with a RF receiver 4204 attached to an antenna 4202 for reception of positioning signals. A controller 4206 receives data from the receiver and logic processing circuits 4210. The logic processing circuits 4210 may include, but are not limited to crosscorrelators, match filters, and FFT circuits. The controller 4206 is in signal communication with, clocks
15 4208, the logic processing circuits 4210, 4210, random access memory 4212, and read-only memory 4214 over an electrical path 4218, such as an electrical bus. The read-only memory 4214 contains a plurality of encoded instructions that are compiled form a high level computer language, such as "C", object oriented language such as "C++" or LISP. The encoded instructions in another implementation may be an interpreted language such
20 as "Perl". The satellite positioning receiver 4200 is shown as a single block. In other implementation, the satellite positioning receiver 4200 may be a multifunction satellite positioning receiver or multimode satellite positioning receiver under the control of the

Expert System 4208. In yet another implementation, the Expert System may reside in a removable storage media, such as compact flash card, smart card, floppy disk, or other type of permanent media.

[0292] Turning to figure 43, the system control and management processes 4300 are shown. The processes 4300 are executed by the controller 4206 and may be monitored and controlled by the Expert System residing in ROM 4214. The operating system or main control loop is the system control and management process 4302. Upon power-up or initialization, the system control and management process starts. Part of the system control and management process 4302 is the expert system process 4303. Initial status data from the satellite positioning system receiver 4200 is accessed by the expert system process 4303 from the startup process 4303 and hardware. Similarly, if the satellite positioning system receiver is reset, the reset process 4306 initiates the reset and the expert system process 4303 receives status data from the other processes and hardware. The interfaces process 4308 enable cross process communication and communication with the hardware. The acquisition process is executed when satellite acquisition is required. The expert system 4303 is notified of the need for acquiring the satellites and allocates memory and processing resources accordingly.

[0293] The signal processing process 4312 processes the positioning signals received by the RF receiver 4204 and works in conjunction with the corsscorelator process 4314. The expert system process 4303 may allocate resources, such as memory, correlators, filters, and processing cycles in order to attain maximum desired performance. The

expert system process 4303 also receives hardware data from the hardware data process 4316 and tracking data from the tracking processes 4318.

[0294] The expert system may be programmed with a plurality of algorithms that monitor the battery life of a mobile device and configures the satellite positioning system receiver 4200 in a way to extend the battery life. Examples of how the expert system would adapt the operation of the satellite positioning system receiver 4200, may include adjusting the number of correlators and filters used to process the positioning signal once acquisition is achieved. Or in another implementation, the expert system may control where the processing of the received data from the positioning signal is processed (i.e. on a host or locally).

[0295] The expert system may also adapt the operation of the satellite positioning system receiver in response to environment, such as enabling the use of aiding data, changing the processing of the received positioning signal to raise or lower the sensitivity of noise in the signal.

[0296] The expert system may be implemented in a layered approach that as shown in figure 44. The design of the satellite positioning system receiver 4200 has a plurality of layers 4400. The first layer is the application layer 4402. The application layer may contain the satellite positioning system 4404 which contains the control and management process 4302.

[0297] The control and management process 4302 communicates with the platform layer 4412 via control process 4410 in the control and management process 4412 of the platform layer 4408. The control process 4410 receives data from other platform layer

processes including the startup 4420, reset 4422, acquisition 4414, tracker 4416, and crosscorrelator 4418. The control process 4410 in the control and management process 4412 of the platform layer 4408 also communicates with the hardware/driver layer 4424 which has the processor 4428 functioning as a controller 4206 and the clocks and timers 4208. The layered approach enables the expert system process in the application layer to communicate and receive data from the other layers and processes in an organized and efficient manner.

[0298] The expert system may execute a plurality of steps 4500 such as described in figure 45. The steps start 4502 with the processing of data for use in search strategy from acquisition and track modules 4504. The interface is then checked for the status of the acquisition 4506. An output message is created if the appropriate time has been reached to acquire data 4508. The receiver is checked to identify if a 100ms boundary has been met or if the measurements are available and resources may be reallocated by the expert system 4510. The commands issued to the interface relating to the satellite vehicle (SV) are checked 4512. The expert system then receives information about the processing of new and updated SV records in order to determine a search strategy 3414. The hardware status (memory, execution, buffers, timers, filters, clocks, correlators) is checked and reported to the expert system 4516. The process steps are then repeated. The process is shown as a simplified control loop, but in other implementations a more advanced adaptive control loop may be used.

[0299] Numerous communication systems rely on receiving one or more radio frequency (RF) signals. As the available frequency bands become more congested, the

numerous radio frequencies used by the different communication systems start to interfere with each other. One type of interference encountered in the different communication systems is carrier wave (CW) interference or more commonly called CW jamming.

5 **[0300]** CW jamming is a source of interference in spread spectrum systems, such as CDMA cellular telephone systems and satellite positioning systems. Spread spectrum communication systems use lower power signals spread across the frequency spectrum and are subject to interference from carrier waves used in other communication systems. The problem of CW jamming is further complicated because of the geographical area
10 covered by spread spectrum system may include the whole Earth, for example the United States' Global Position System (GPS).

[0301] Attempts to eliminate CW jamming signals have occurred by regulating frequency use and by adding dedicated circuitry to radio receivers. The added circuitry being included in radio receivers often result in additional power consumption and
15 expense.

[0302] Therefore, there is a need for methods and systems for identifying and removing CW jamming signals that overcomes the disadvantages set forth above, and others previously experienced.

[0303] Systems consistent with the present invention provide a receiver that is
20 capable of receiving a spread spectrum signal that contains a CW jamming signal along with a weak signal. The signal is processed with a crosscorrelator that enables the CW jamming signal to be identified, tracked, and reproduced. The replicated CW jamming

signal is subtracted from the received signal after demodulation, thus enabling the weak signal to be processed.

[0304] Other systems, methods, features and advantages of the invention will be or will become apparent to one with skill in the art upon examination of the following
5 figures and detailed description. It is intended that all such additional systems, methods, features and advantages be included within this description, be within the scope of the invention, and be protected by the accompanying claims.

[0305] Turning first to Figure 42, a flow diagram 4200 depicting strong signal cancellation in a weak spread spectrum signal using crosscorrelation is shown. A
10 strong/weak or near/far signal isolation provided by a spread spectrum, pseudo random number (PRN) code family such as used in CDMA spread spectrum systems is dependent upon the crosscorrelation between the various code members of the family. In the case of a satellite positioning system, such as GPS, the isolation of two signals at the same frequency (or multiples of the code repetition rate, in this case 1 KHz) is about 21 to 23
15 dB. If the relative strengths of two signals differ by more than this limit, the weaker signal cannot be discriminated using only the spreading code. A method of removing the effects of the stronger signal may be applied if the weaker signal is to be tracked.

[0306] The crosscorrelation effect is at its maximum when the relative Doppler frequency offset between the relatively strong and weak signals is an integer multiple of 1
20 KHz in the case of coarse acquisition code (C/A) in the GPS signals. A general solution to the problem of tracking a weak signal spread spectrum signal in the presence of a stronger spread spectrum signal is based on the premise that all aspects of the strong

signal's interference can either be measured or calculated in order to remove it from the weaker signal. The solution can be implemented in any multi-channel receiver having the ability to control a channel's frequency and phase as well as selecting the desired spreading code and setting that code's phase position. The receiver typically employs
5 two channels, one to track the weak signal and one to track the interfering strong signal. However, the channel that is used to track the strong signal is not required if the characteristics such as power, code phase and frequency of the strong signal can be obtained or accurately estimated by alternate means.

[0307] As shown generally in FIG. 42, the procedure starts 4202 with the strong
10 signal being acquired 4204, such as by tracking the strong signal in a first channel of the receiver. The channel provides a measurement of the signal strength of the strong signal along with the phase of the carrier signal and the spreading code. Additional channels may be used to track additional strong signals (not shown in FIG. 42).

[0308] The code phase of the spreading code of the weaker signal, along with its
15 received frequency and signal phase, are predicted 4206 based on the 50Hz navigation data code data (D) by methods known in the art. A second channel in the receiver is dedicated to receiving the compound carrier signal and tracking 4208 the predicted weak signal component.

[0309] The second receiver channel correlates the incoming signal with the second
20 code at the predicted frequency and signal phase. The resulting in-phase and quadrature (I,Q) measurements contain both the weak signal and the strong signal, each spread by their unique code. Correlation by multiplication of the replica code for the second signal,

Code2R, with the incoming signal yields the product $\text{Code2R} * (\text{weak2} * \text{Code2} + \text{StrongX} * \text{CodeX} + \dots)$ where weak2 is the power of weak signal 2, Code2 is the actual code for satellite 2 broadcasting the weak signal 2, StrongX ($X=1, 3, 4, \dots$) is the power of strong signal X, and CodeX is the actual code for satellite X contained in the signal. The product $\text{Code2R} * \text{Code2}$ is the autocorrelation of the received code 2 and the replica code 2. The autocorrelation function has a value of 1 if the replica code is aligned with the received code. This crosscorrelation of replica code 2 with code X ($\text{Code2R} * \text{CodeX}$) is next computed 4210 to be removed from the compound signal.

10 [0310] Code1 and Code2 are both members of a PRN code family, and their autocorrelation and crosscorrelation properties are known. It is therefore possible to calculate the crosscorrelation of the two codes at their respective phases by simply multiplying each bit of Code1 by the corresponding (in time) bit of Code2 to produce their crosscorrelation value. Since there may be a relative Doppler frequency offset
15 between the two codes, the phase of the codes will process past one another over time and create a new crosscorrelation function. For the GPS system the greatest delta code Doppler typically encountered is about plus or minus 9KHz which is equivalent to six code chips per second (1540 carrier cycles per code chip), and thus the maximum recalculation rate of the crosscorrelation value is roughly 6 times per second.

20 [0311] The maximum crosscorrelation occurs at a frequency offset of zero with peaks occurring at intervals of 1000Hz. There is an attenuation of the crosscorrelation as the frequency offset moves away from zero. This attenuation follows the well known

sin(x)/x curve. If 10ms measurements are used for tracking or acquisition, the attenuation factor would be equal to $\sin(\Delta\text{freq} \cdot \pi / 100 \text{ Hz}) / (\Delta\text{freq} \cdot \pi / 100 \text{ Hz})$. This produces an attenuation of -10dB at about a 75Hz delta frequency. Other local peaks in the sin(x)/x curve (i.e. locally minimum attenuation) occur at 150Hz and 250Hz with
5 attenuations of -13.5dB and -18dB, respectively. This implies that for a desired strong signal suppression of 10dB, only the first lobe of the sin(x)/x function need be considered; however, should additional suppression be desired, the entire curve may be considered.

[0312] The next step entails computing 4212, for each strong signal, the product of
10 the strong signal amplitude and the calculated frequency and time domain (code phase) crosscorrelation. The weak signal is finally extracted by subtracting 4214 this product from the compound signal and processing is complete 4216. The weak signal thus extracted is subsequently processed in the receiver circuitry as known in the art.

[0313] The in-phase (I) and quadrature amplitude (Q) of each strong signal is
15 obtained by measurement in each strong signal's own individual receiver channel or by estimation through independent means. Because the strong signal is being actively tracked by the receiver's phase lock loops, the phase of the strong signal is presumed to be near zero radians and thus nearly all the signal power is in the in-phase portion.

[0314] A signal comprising a strong signal S1 modulated with a first code Code1
20 summed with a weak signal w2 modulated with a second code Code2 produces $(S1 \cdot \text{code1} + w2 \cdot \text{Code2})$. The sum of the two signals is correlated with a replica of the second code Code2R to produce $\Sigma\{\text{Code2R} \cdot (S1 \cdot \text{Code1} + w2 \cdot \text{Code2})\}$, where the sum Σ

includes all chips of the PRN code used to modulate the weak signal w_2 . The autocorrelation of a code with itself is 1 so the preceding equation can be rewritten as $\Sigma\{S_1 * \text{Code}_1 * \text{Code}_2 + w_2\}$. It can be seen that in order to obtain w_2 $S_1 * \text{Code}_1 * \text{Code}_2$ is removed. Since we know Code_1 and Code_2 , we can easily calculate their crosscorrelation. This leaves us to estimate the value of S_1 that can be done by independently tracking the strong signal on a separate channel, or by any other convenient means. This computed value of $S_1 * \text{Code}_1 * \text{Code}_2$ would be sufficient if the strong signal S_1 and the weak signal w_2 were at the same frequency. The two signals are received at different frequencies, however, due to the Doppler Effect as well as other factors enumerated previously.

[0315] We know that strength of the crosscorrelation varies with the difference between these frequencies in a $\sin(x)/x$ relationship. We may therefore calculate an attenuation factor based on the difference in frequency between the strong and the weak signal and apply it to the computed crosscorrelation. Furthermore, if more than one strong signal is present, an attenuation factor is computed for each strong signal.

[0316] The code dependent portion of the crosscorrelation factor is computed from the known relative states of the PRN code generators to predict the crosscorrelation between a strong signal of unit power and zero frequency offset, and a weak signal. This factor is multiplied by the amplitude of the corresponding strong signal and adjusted for frequency attenuation before it is subtracted from the composite signal.

[0317] The various Gold codes used to modulate the PRN signals are all derived from a two code sequence G_1 and G_2 where the bits of the two code sequences are combined

through an XOR operation after G2 has been offset some number of bits relative to G1 depending on the Gold code selected. It is known that an XOR operation using binary numbers is mathematically equivalent to multiplication of ± 1 . This allows expressing the equations below in term of products of ± 1 while in reality the implementation could be
5 with binary numbers with XORs.

[0318] The correlation between two C/A codes can generally be expressed as:

[0319]
$$\sum \text{Sat1G1(I)} * \text{Sat1G2(I)} * \text{Sat2G1(I-offset)} * \text{Sat2G2(I-offset)} * e^{-j\Delta\theta I}$$

[0320] where

[0321] I=Summation index ranges from 0 to 1022.

10 [0322] Sat1G1(I)=Value of satellite 1's G1 coder chip at state I. Possible values are ± 1 .

[0323] Sat1G2(I)=Value of satellite 1's G2 coder chip at state I. Possible values are ± 1 .

[0324] Sat2G1(I)=Value of satellite 2's G1 coder chip at state I. Possible values
15 are ± 1 .

[0325] Sat2G2(I)=Value of satellite 2's G2 coder chip at state I. Possible values are ± 1 .

[0326] offset=time difference between the satellite 1 and 2 in units of chips
 $\Delta\theta$ *Phase change per chip between satellite 1 and 2 in radians.

20 [0327] It should be noted that when the difference I-offset is less than 0, 1023 is added to the difference to maintain the value in the range of 0 to 1022. In other words,

the domain of the functions returning coder chip states is limited to the range of 0 to 1022.

[0328] The computation time required to compute the 1023 bit-by-bit correlations can be accelerated by making use of standard controller instructions that perform 8, 16 or 32 bit-wise XORs with a single controller instruction. The following will demonstrate the method of computing eight chips in parallel. Those skilled in the art will immediately recognize that the scheme can be easily modified to accommodate some other convenient number of bits per controller XOR operation.

[0329] The 1023 states of G1 and G2 may be stored linearly in permanent memory. Thus, it is possible to quickly gather 8, 16, 32 or some other convenient number of bits with a single controller load instruction by computing the address of the desired chip and the shift required to align it. Thirty-two bits is a particularly convenient number because 31 divides 1023 evenly. The current implementation thus reads 32 bits at a time and uses 31 of them at a time for each of 33 intervals that span the 1023 chips of the C/A code. The 31 bit sums are broken into four parts of 8, 8, 8, and 7 bits, and each 7 or 8 bit sum is multiplied by $e^{-j\Delta\theta I}$ where I changes by 7.75 chips for each part. The form of the sum is:

[0330]
$$\begin{aligned} & \Sigma(e^{j\Delta\theta I} * 31 * \Sigma(\text{Sat1G1}(I*31+J) * \text{Sat1G2}(I*31+J) * \text{Sat2G1}(I*31+J\text{offset}) * \text{Sat2G} \\ & 2(I*31*J\text{-offset})) + e^{-j\Delta\theta(I*31+7.75)} * \Sigma(\text{Sat1G1}(I*31+J+8) * \text{Sat1G2}(I*31+J+8) * \text{Sat2G1}(I*31+J+8\text{-offset}) \\ & * \text{Sat2G2}(I*31+J+8\text{-offset})) + e^{-j\Delta\theta(I*31+15.5)} * \Sigma(\text{Sat1G1}(I*31+J+16) * \text{Sat1G2}(I*31+J+16) \\ & * \text{Sat2G1}(I*31+J+16\text{-offset}) * \text{Sat2G2}(I*31+J+16\text{-offset})) + e^{-j\Delta\theta(I*31+23.25)} * \Sigma(\text{Sat1G1}(I*31+J+24) * \text{Sat1G2}(I*31+J+24) * \text{Sat2G1}(I*31+J+24\text{-offset}) * \text{Sat2G2}(I*31+J+24\text{-offset})) \end{aligned}$$

$$j\Delta\theta(I*31+23.25)*\Sigma(\text{Sat1G1}(I*31+J+24) \quad * \text{Sat1G2}(I*31+J+24)*\text{Sat2G1}(I*31+J+24- \\ \text{offset})*\text{Sat2G2}(I*31+J+24-\text{offset})))$$

[0331] where

[0332] I=Outer index ranges from 0 to 32

5 [0333] J=Inner index ranges from 0 to 7 for the first three sums and from 0 to 6
for the last sum. The inner sums are computed in parallel by using a 32 bit
word that contains all 31 bits and using bitwise XOR to perform the
multiplications and shifting and adding to sum the 1 bit products.

[0334] Note that all of the multiplications of the G1 and G2 codes in the above
10 equation are implemented by bit-wise XOR instructions. The above algorithm is in error
by at most -17 dB from an exact computation, and requires about 6000 controller
operations to complete.

[0335] Periodically, as needed, the code dependent crosscorrelation factors are
computed for all strong and weak signal pairs with small frequency differences, i.e.
15 frequency differences that could cause strong-weak crosscorrelation interference. In the
current implementation strong signals are those with $C/N_0 > 40$ dB and weak signals are
those with $C/N_0 > 30$ dB. Because 10ms integrations of I and Q measurements are used
by the code and phase tracking loops, the maximum "significant" frequency difference
(modulo 1000Hz) is 90Hz. In the preferred embodiment the code dependent cross
20 correlation factor for each possibly interfering pair of signals is computed for each of the
measurements that might potentially be used by the tracking and signal processing
algorithms. For example, if early, punctual and late measurements are used by the

tracking loops, the correlation factors for each of these code alignments is computed and stored in the tables.

[0336] These tables only need be updated at a 10Hz rate because the maximum Doppler difference is less than 9KHz or less than six chips per second. In addition to
5 maintaining the crosscorrelation table, the frequency attenuation of the crosscorrelation due to the frequency difference is computed at the 10Hz rate. The attenuation can be expressed as:

[0337] $\text{Frequency Attenuation} = \sin(\Delta F \bmod 1000 * \pi / 100) / (\Delta F \bmod 1000 * \pi / 100)$

[0338] where

10 [0339] ΔF = Frequency difference between a strong and weak signal in Hz
Mod = modulo offset to give a range of -500 Hz to +500 Hz

[0340] The attenuation only needs to be recomputed if the frequency difference changes by more than 5Hz.

[0341] An estimate of the phase and amplitude of the strong signal is required to
15 remove the crosscorrelation. The method used in the preferred embodiment is to track the strong signal on its own dedicated channel and collect the I, Q measurements output over the exact same interval that the weak signal I, Q samples are taken. The known phase and frequency of the replica signal that is used to track the strong signal is an excellent approximation of the actual phase and frequency of the strong signal.
20 Furthermore, because the strong signal is in phase lock, the magnitude of the I measurement provides a good approximation of the amplitude of the strong signal. Finally, the bi-phase modulation of the strong signal data bits D may cause the phase of

the strong signal to rotate 180 degrees whenever the data bits transition from a 1 to 0 or from a 0 to 1. In the current implementation, the phase of the strong signal is corrected by adding 180 degrees to the phase of the replica signal whenever the sign of the I measurement for the strong signal is negative.

5 [0342] Every 10ms a new set of I, Q correlation data is available from the channel assigned to track the weak signal. The tables of crosscorrelation factors are checked to predict the presence of any interfering strong signals. If strong signals are predicted, the following subtraction is performed to remove the strong signal crosscorrelations:

[0343] $\text{FirstCodeOffset} = \text{WeakCodeState} - \text{StrongCodeState} - \text{StrongDoppler} * \Delta T -$
10 $\text{TableEntry0CodeState}$

[0344] $\text{DeltaPhase} = \text{WeakCarrierPhase} - \text{StrongCarrierPhase} - \text{StrongDoppler} * \Delta T$
 $+ \text{DeltaKHz} * \text{StrongCodeState}$

[0345] $\text{FirstPhase} = \text{FirstCorrelationPhase} + \text{DeltaPhase}$

[0346] $\text{SecondPhase} = \text{SecondCorrelationPhase} + \text{DeltaPhase}$

15 [0347] $\text{FirstMag} = \text{FirstCorrelationMag} + \text{FirstCodeOffsetFraction} * \text{StrongI}$
 $* \text{FrequencyAttenuation}$

[0348] $\text{SecondMag} = \text{SecondCorrelationMag} * (1 - \text{FirstCodeOffsetFraction})$
 $* \text{StrongI} * \text{FrequencyAttenuation}$

[0349] $\text{CorrectedWeakIQ} = \text{WeakIQ} - \text{FirstMag} * e^{-j\text{FirstPhase}} - \text{SecondMag} * e^{-j\text{SecondPhase}}$

20 [0350] where

[0351] $\text{WeakCodeState} = \text{Code state of last output to the weak signals channel}$

- [0352] StrongCodeState=Code state of last output to the strong signals channel
- [0353] StrongDoppler=Doppler of last output to the strong signals channel
 ΔT =Difference in time between outputs to the weak and strong channels
- [0354] TableEntry0CodeState=Code state difference of the first element of the
5 crosscorrelation table.
- [0355] WeakCarrierPhase=Carrier phase angle of last output to the weak signal
channel
- [0356] StrongCarrierPhase=Carrier phase angle of last output to the strong signal
channel
- 10 [0357] DeltaKHz=Nearest integer multiple of 1 KHz of the difference between
the weak and strong channels Doppler. In units of KHz.
- [0358] FirstCorrelationPhase=Phase entry in the crosscorrelation table for the
chip indicated by FirstCodeOffset
- [0359] SecondCorrelationPhase=Phase entry in the crosscorrelation table for the
15 chip indicated by FirstCodeOffset +1 chip.
- [0360] FirstCorrelationMag=Magnitude entry in the crosscorrelation table for the
chip indicated by FirstCodeOffset.
- [0361] SecondCorrelationMag=Magnitude entry in the crosscorrelation table for
the chip indicated by FirstCodeOffset+1 chip.
- 20 [0362] FirstCodeOffsetFraction=Fraction of a chip in FirstCodeOffset.
- [0363] StrongI=Absolute value of I correlation from the strong channel.
- [0364] FrequencyAttenuation=Attenuation due to frequency offset.

[0365] WeakIQ=IQ correlation from the weak signal's channel.

[0366] CorrectedWeakIQ=IQ correlation corrected for crosscorrelation from the strong signal. CorrectedWeakIQ is computed for the early, on time, and late correlators by shifting the FirstCodeOffset appropriately, such as by half a chip each. These modified correlations are then used normally in the carrier and code tracking software for the weak signal. The algorithm attenuates the crosscorrelation by at least 10 dB without attenuating the weak signal, and is repeated for each strong signal that may be interfering with the weak signal.

10 [0367] While figure 42 has described crosscorrelation to cancel a strong signal having a PRN from a weak signal having different PRN, the procedure may be modified to cancel a CW jamming signal and enhance a weak signal having a PRN. In figure 43, a signal-processing diagram for identifying and removing CW jamming signals where the cancelled signal of figure 42 may be a CW jamming signal is shown. The receiver 4300
15 receives a spread spectrum signals at antenna 4302. The crosscorrelator 4304 is placed in a mode to identify CW jamming signals. In that mode, the crosscorrelator employs all ones for a PRN code. The crosscorrelator 4304 may have a signal processor 4306 and a match filter 4308 that despreads the spread spectrum signal. The resultant signal is the CW jamming signal (the strong signal) and is tracked in the tracker 4310. An example of
20 the tracker 4310 in the current implementation may be a phase lock loop circuit.

[0368] The identified CW jamming signal is then used to generate a replica CW jamming signal 4312. The replica CW jamming signal may be created using a signal

processor 4314 and a match filter 4316. In other implementations, other approaches to generating a desired signal may be employed. Examples of other approaches include, but are not limited to voltage controlled oscillators, digital signal processing, analog signal processing.

5 [0369] The spread spectrum signal received by receiver 4300 is then processed by a crosscorrelator 4314 with despreading codes rather than the all ones. The received weak signal in the spread spectrum signal is demodulated by the signal processor in the crosscorrelator 4318. The generated CW jamming signal is canceled from the weak signal by a canceller 4320. The resultant signal is then tracked by a tracker 4322. The
10 CW jamming signal that is being tracked by tracker 4310 is further processed to remove the desired weak signal from the tracked CW jamming signal by signal combiner 4324.

[0370] The identification of the CW jamming signal has been described as occurring with three crosscorrelators 4304, 4318 and 4312. But in practice, only one crosscorrelator may be used allowing the crosscorrelator to distinguish between PRN
15 codes and to identify and remove CW jamming signals. Thus, the generation of a replica of the unwanted signal (CW jamming signal) occurs during the demodulation of the wanted signal.

[0371] Turning to figure 44, a block diagram of electrical components of figure 43 is shown. The spread spectrum signal containing the weak signal and the CW jamming
20 signal are received at the receiver 4300 via antenna 4302. The spread spectrum signal is demodulated and the CW jammer signal is filtered by the jammer filter 4402. The resulting CW jamming signal is tracked by jammer tracker 4404. The tracked jamming

signal is then replicated by a jammer replica wave generator 4406 as a replicated CW jamming signal. The CW signal has the characteristic of a constant phase, therefore the replicated CW jamming signal is scaled and rotated 4408 to an appropriate phase to be subtracted from the demodulated weak signal. The weak signal is demodulated by a demodulator 4410. The phase and magnitude replicated CW jamming signal is then subtracted from the weak signal from the demodulator 4402 by a signal canceller 4412. The resulting weak signal is then the signal is tracked by a track control circuit 4414. The track control circuit 314 outputs the weak signal to the demodulator 4410 and two signal combiners 4416 and 4418. The weak signal is subtracted from the CW jammer signal by the signal combiner 4416 and the resulting signal is used to scale and rotate the replicated CW jamming signal. Similarly, the weak signal is subtracted from the CW jamming signal by signal combiner 4418 in order to provide a more accurate CW jamming signal to be replicated by the jammer replica wave generator 4406.

[0372] Even though the current implementation has been described with CW jamming signals in general, one skilled in the art would recognize that specific known signals such as IS-95 pilot signals, Galileo signals, or other known carriers may be identified, tracked and remove from a demodulated desired signal. Unlike other known methods of identifying and removing CW jamming signals prior to signal processing or at the front end of a receiver, the current approach identifies and removes the CW jamming signal at the back end. This is advantageous when desired signals are received at low signal strength, such as in CDMA or satellite positioning system. The signal

strength is not sufficient to support preprocessing of the signal while maintaining the desired weak signals.

[0373] Turning to figure 45, a flow diagram 4500 of the identifying and removing CW jamming signals is shown. The procedure starts 4502 with a receiver 4300 receiving
5 a spread spectrum signal 4504. The signal may be filtered and the CW jamming signal is identified 406 using a crosscorrelator. Once the CW jamming signal is identified 4506, it is tracked 4508. The tracked CW jamming signal is then replicated 4510 by a jammer replica wave generator 4406. The replicated CW jammer signal is then subtracted from the received signal 4512. The CW jamming signal has been removed from the received
10 signal and processing is complete 4514.

[0374] In figure 50, an exemplary high level implementation of a tracking system in for strong and medium signal operation is shown. The Hardware Tracking Loop ("HWTL") operates at Pre Detection Integration ("PDI") rate. The implementation is done in Subsystem 3. Implementation details available in HWTL ASIC Design Specs.
15 The hardware ("HW") may not transition on its own from Acquisition mode to tracking mode and activate HWTL autonomously. The mode transition to activate HWTL will be under software ("SW") control at the successful completion of Acquisition process.

[0375] Typically the inputs from the hardware will be:

[0376] I_e , Q_e and I_l , Q_l , f_{+e} and f_{-e} , f_{+l} and f_{-l} for early and late taps;

20 [0377] I_e = In Phase Correlation for early tap;

[0378] Q_e = Quad Phase Correlation for early tap;

[0379] I_l = In Phase Correlation for late tap;

[0380] Q_l = Quad Phase Correlation for late tap;

[0381] f_{+e} = Correlation Magnitude for +1 frequency bin offset for early tap;

[0382] f_{-e} = Correlation Magnitude for -1 frequency bin offset for early tap;

[0383] f_{+l} = Correlation Magnitude for +1 frequency bin offset for late tap;

5 [0384] f_{-l} = Correlation Magnitude for -1 frequency bin offset for late tap; and

[0385] Autoscale Values from HW for normalization purposes.

[0386] As an example the following states may be initialized utilizing information from Acquisition: 1. Code Phase; 2. Carrier Frequency; 3. Carrier Phase; and 4. Filtered Signal amplitude estimate used to normalize the tracking loops ("S_Gain") (if amplitude
10 estimate is available).

[0387] In this example, the code discriminator may be

[0388] $D = |e| - |l|$,

[0389] $|e| = \sqrt{I_e * I_e + Q_e * Q_e}$, and

[0390] $|l| = \sqrt{I_l * I_l + Q_l * Q_l}$,

15 [0391] and the carrier phase discriminator may be

[0392] $\phi = \text{sign}(I_p) * Q_p$,

[0393] $I_p = I_e + I_l$, and

[0394] $Q_p = Q_e + Q_l$,

[0395] and the carrier frequency discriminator may be

20 [0396] $\delta f = f_+ - f_-$,

[0397] $f_+ = f_{+e} + f_{+l}$, and

[0398] $f_- = f_{-e} + f_{-l}$

[0399] where all computation is done using magnitudes or magnitude approximations.

[0400] The subsystem 2 processing may be described by

[0401] Carrier Phase += Carrier Frequency,

5 [0402] Carrier Frequency += Carrier Frequency Rate, and

[0403] Code Phase += Delta Carrier Phase.

[0404] As an example, the tracking loop equations may be described by

[0405] Carrier Phase += $K1 * \phi + K2 * \delta f + \text{Aid } 1$,

[0406] Carrier Frequency += $K3 * \phi + K4 * \delta f + \text{Aid } 2$,

10 [0407] Carrier Frequency Rate += $K5 * \phi + K6 * \delta f + \text{Aid } 3$, and

[0408] Code Phase += $K7 * D + \text{Aid } 4$.

[0409] The gains K1 through K7 may be input from SW at 100 ms rates. The HWTL mode transitions may be controlled through the values of these gains.

15 [0410] As a general example, the tracking modes and mode transition may be described by the following description.

[0411] Controlled by SW at 100 ms rate through setting the gains K1-K7.

[0412] In Init Mode, the Tracking Loops may be initialized with a wide bandwidth (preferably at approximately 1/10 of iteration rate PDI) frequency Loop. Code Loop is initialized with wide bandwidth (preferably at 5 Hz * Acq Error Estimate) Transition to
20 Narrow Bandwidth Frequency Loop if filtered frequency error estimate < threshold1.
Transition to wide carrier phase loop if filtered frequency error estimate < threshold2.
Recommended PDI is 4 ms. Preferably the FFT mode is 8 point.

[0413] In the narrow frequency loop, after entry from Init Mode. The Tracking Loops may be set to narrow bandwidth frequency Loop (preferably at approximately 1/20 of PDI rate). Narrow Code Loop (preferably at approximately bandwidth 1/3 Hz) if estimated code error < 0.1 chips Transition to carrier phase loop if filtered frequency error estimate < threshold1. Transition to Init Mode if filtered frequency error estimate > threshold2. Preferably the PDI is 4 ms and the FFT mode is 8 point.

[0414] In the wide carrier phase loop, after entry from Init Mode or Narrow Frequency Loop. The Tracking Loop may be set to wide carrier (preferably at approximately a bandwidth 1/4 of PDI rate) and narrow frequency combined loop using K1-K7. Narrow Code Loop (preferably at approximately a bandwidth 1/3 Hz) if estimated code error < 0.1 chips Transition to Narrow Frequency Loop if (carrier Phase lock lost and filtered frequency error estimate < threshold1). Transition to Init Mode if (carrier Phase lock lost and filtered frequency error estimate < threshold2). Transition to Narrow Carrier Loop if filtered carrier phase error < Threshold. Preferably at approximately PDI 4 ms in absence of bit sync and 20 ms in presence of bitsynch and with a FFT mode of 20 point.

[0415] In the Narrow Carrier Phase Loop, after entry from Wide Carrier Phase Loop. Tracking Loop set to narrow carrier (preferably at approximately a bandwidth 1/10 of PDI rate) and narrow frequency combined loop using K1-K7. Narrow Code Loop (preferably at approximately a bandwidth 1/3 Hz) if estimated code error < 0.1 chips Transition to Narrow Frequency Loop if (carrier Phase lock lost and filtered frequency error estimate < threshold1). Transition to Init Mode if (carrier Phase lock lost and

filtered frequency error estimate < threshold2). Preferably at approximately a PDI 4 ms in absence of bit sync and 20 ms in presence of bitsynch and FFT mode is 20 point.

[0416] In the following Mode Transition Table, the columns are the source modes and rows are the destination modes. The entry in the table describes the transition condition.

[0417] Mode Transition Table 34

	Init	Narrow Freq	Wide Carrier	Narrow Carrier
Init	X	filtered frequency error estimate < threshold1	filtered frequency error estimate < threshold2.	X
Narrow Freq	filtered frequency error estimate > threshold2	X	filtered frequency error estimate < threshold1	X
Wide Carrier	Mode if (carrier Phase lock lost and filtered frequency error estimate > threshold2).	if (carrier Phase lock lost and filtered frequency error estimate > threshold1).	X	X
Narrow Carrier	if (carrier Phase lock lost and filtered frequency error estimate > threshold2).	if (carrier Phase lock lost and filtered frequency error estimate > threshold1	X	X

[0418] Code Loop Transition Table 35

	Init (High Bandwidth)	Low Bandwidth
Init (High Bandwidth)	X	estimated code error < 0.1 chips
Low Bandwidth	estimated code error > 0.1 chips	X

[0419] All modes may transition out of the tracking system if code and frequency lock are lost and acquisition may commence.

5 **[0420]** Figure 51 shows the hardware update. At the end of processing for the Subsystem 3 the Carrier Phase, Carrier Frequency, Carrier Frequency Rate and Code Phase is updated in the Subsystem 2 state.

[0421] The code state has a coarse resolution and the code phase equations can accumulate small code phase updates and when the code lsb is reached the code state in
10 subsystem 2 will be updated. This can be accomplished using the following mechanization. This mechanization describes an approximation for the normalized early – late triggering the changes to the code state. N is the number of LSBs in the code state to change.

[0422] AGC Normalization/ Auto Scale

15 **[0423]** $S_Gain(t+1) = \text{Alpha} * S_Gain + \text{Beta} * (|I_p| + |Q_p|)$

[0424] The S_Gain is used to normalize the Loop Coefficients. "t" is the time index. S_Gain may also be used to Normalize the early minus late output to determine the update to the code state in HW.

[0425] The SWTL will be activated at the same time as the HWTL by SW control at the end of successful acquisition process. The SWTL is operated at 100 ms rate. The input consists of buffered reports at the PDI rate and reports on NCO states. The SW is expected to compute more accurate error estimates use aiding from NAV and or outside sources and construct single correction every 100 ms to adjust the HWTL. In special cases the HWTL may also be disabled and SWTL may operate autonomously generating corrections to the HW NCOs at rates different than 100 ms. The HW will provide interrupt at higher rates and the SWTL will operate at rates faster than 100 ms in such cases. The HWTL can be disabled using a single control bit for each channel independently.

[0426] Inputs From HW

[0427] NCS Buffer and Track History Buffer at 100ms rate. Contents: I and Q correlation outputs at each PDI for various offsets, Noise sums for PDIs, AutoScale Values, NCO States sampled every PDI, Time Mark representing time of Measurement Report.

[0428] State Initialization

[0429] The following states will be initialized using information from Acquisition.

[0430] 1. Code Phase;

[0431] 2. Carrier Frequency;

20 [0432] 3. Carrier Phase; and

[0433] 4. S_Gain (if amplitude estimate is available)

[0434] Code Discriminator

[0435] The code discriminator is given by the following equation.

[0436] $D = |E| - |L|$

[0437] $|E| = \text{Alpha} * |Ee| + \text{Beta} * |Ee+|$

[0438] $|L| = \text{Alpha} * |Le| + \text{Beta} * |Le+|$

5 [0439] where Ee and Le are the early and late I and Q values for one tap off from prompt and Ee+ and Le+ are the early and late I and Q values for two tap off from prompt.

[0440] Filtered Code Error = Alpha * Filtered Code Error + Beta * D.

[0441] Carrier Phase Discriminator

10 [0442] $\phi = \arctan(Qp, Ip)$

[0443] The ϕ is computed for each PDI and a $d\phi$ is maintained in software.

[0444] Carrier Frequency Discriminator

[0445] $\delta f = \text{Alpha} * (f_+ - f_-) + \text{Beta} * (Ip(t+1)*Qp(t) - Ip(t)*Qp(t+1))$

[0446] where t is the time index.

15 [0447] Tracking Equations

[0448] The tracking equations will be iterated at PDI rate. The corrections back into HW will be generated at 100 ms.

[0449] Phase Tracking Loop

[0450] Carrier Phase += Carrier Frequency + K1 * $d\phi$ + Aid 1

20 [0451] Carrier Frequency += Carrier Frequency Rate + K3 * $d\phi$ + Aid 2

[0452] Carrier Frequency Rate += K5 * ϕ + Aid 3

[0453] Frequency Tracking Loop

- [0454] Carrier Frequency += Carrier Frequency Rate + $K4 * \delta f$ + Aid 2
- [0455] Carrier Frequency Rate += $K6 * \delta f$ + Aid 3
- [0456] Code Loop
- [0457] Code Phase += Scale*(Carrier Phase (t) – Carrier Phase(t-1)) $K7 * D$ + Aid4
- 5 [0458] $K7$ initialized to K_{max} (high bandwidth value) and updated as follows
- [0459] $K7(t+1) = \text{Alpha} * K7 + \text{Beta} * \text{Code Error Estimate}$
- [0460] If $K7 > K_{max}$ $K7 = K_{max}$
- [0461] If $K7 < K_{min}$ $K7 = K_{min}$
- [0462] The tracking loop mode transition is controlled through the gains $K1$ through
- 10 $K7$.
- [0463] AGC Normalization
- [0464] $S_Gain(t+1) = S_Gain_Rate * \text{Gamma} + \text{Alpha} * S_Gain + \text{Beta} * (|Ip| +$
 $|Qp|)$
- [0465] $S_Gain_Rate(t+1) = \text{Alpha} * S_Gain_Rate + \text{Beta} * [(|Ip(t)| + |Qp(t)|) - (|Ip(t-$
15 $1)| + |Qp(t-1)|)]$
- [0466] The S_Gain and the S_Gain Rate represent the estimate of magnitude of the
signal and the rate of magnitude change for the signal.
- [0467] The S_Gain will be used to normalize the tracking loop equation gains.
- [0468] Loss Of Lock Detectors
- 20 [0469] Code Lock
- [0470] Compute the Signal to Noise Ratio at 100 ms rate. The computation may be
performed as follows:

[0471] Noise Power = $\text{Sum}(I_n * I_n + Q_n * Q_n)$

[0472] The sum is carried over 100 ms time period. I_n and Q_n are I and Q outputs from Noise sum in the HW reports at PDI rate.

[0473] Signal Power = $\text{Sum}(I_p * I_p + Q_p * Q_p)$

5 [0474] The sum is carried over 100 ms time period.

[0475] Signal to Noise Ratio = Signal Power / Noise Power

[0476] Filtered SNR = $\text{Alpha} * \text{Filtered SNR} + \text{Beta} * \text{Signal to Noise Ratio} + \text{Gamma} * \text{SNR_Rate}$

[0477] $\text{SNR_Rate} = \text{Alpha} * \text{SNR_Rate} + \text{Beta} * [\text{Signal to Noise Ratio}(t) - \text{Signal to}$

10 Noise Ratio (t-1)]

[0478] If Filtered SNR < Threshold declare Loss Of Lock

[0479] Carrier Phase Lock

[0480] Estimate the filtered phase error using a 2 Quad arctan function

[0481] $\phi = \arctan(Q_p, I_p)$

15 [0482] computed every PDI

[0483] Filtered $\phi = \text{Alpha} * \text{Filtered } \phi + \text{Beta} * \phi$

[0484] If Filtered $\phi > \text{Threshold}$ declare loss of lock.

[0485] Carrier Frequency Lock

[0486] Estimate Filtered Frequency Error

20 [0487] $\delta f = \text{Alpha} * (f_+ - f_-) + \text{Beta} * (I_p(t+1) * Q_p(t) - I_p(t) * Q_p(t+1))$

[0488] the δf is computed at the PDI Rate.

[0489] Filtered $\delta f = \text{Alpha} * \text{Filtered } \delta f + \text{Beta} * \delta f$

- [0490] If Filtered $\delta f < \text{Threshold}$ declare loss of lock.
- [0491] The update to the HW may be in the form of Aid to the HW Tracking Loop equations. This may be implemented every 100 ms as input from the Software. The Aid may be computed by using the following generic equations.
- 5 [0492] SW to HW Aid = Software Estimate – HW Estimate
- [0493] If the HWTL is disabled the HW Estimate = 0
- [0494] For the phase and frequency and rates. The HW will implement the aid one time only when the SW writes into appropriate registers.
- [0495] Measurement Report Processing
- 10 [0496] The Pseudorange and Rate measurements may be constructed as per the description in the HW to SW interface.
- [0497] The Bit Synch may operate at 20-100 ms rates and may be implemented in SW.
- [0498] Inputs
- 15 [0499] The HW may provide 20 ms accumulations (PDI) for 20 offsets (programmable). The Input Array histogram[20] may consist of 20 ms PDI power accumulations.
- [0500] The Histogram may be accumulated using the following equation for each SV
- [0501] $\text{AccumHistogram}[20] += \text{Histogram}[20]$
- 20 [0502] If offset information is available accum Histograms from multiple SVs may be accumulated using the following equation.
- [0503] $\text{Accum HistogramMultiple SV}[20] += \text{AccumHistogram}[20 + \text{SV Offset}]$

- [0504] First example approach:
- [0505] The histogram peak and second peak detected.
- [0506] If ((Histogram Peak > Threshold) && (Histogram Peak – Histogram Second Peak) > Threshold) Bit Synch Complete Output Bit Location
- 5 [0507] Else continue accumulation.
- [0508] Second example approach:
- [0509] Compute cost function by correlating a triangle 20 ms wide to the accumulated histogram.
- [0510] Cost Function(offset) = Sum(triangle(ms) * Accum Histogram(ms-offset))
- 10 [0511] The sum is carried over the 20 values of ms. The max for the cost function(offset) will provide the bit synch offset.
- [0512] If [max(cost function(offset) > Threshold) && {max(cost function(offset)) – second max(cost function(offset))} > Threshold)] declare bit synch success.
- [0513] Data Demodulation: The Data Demodulation may operate at 20-10 ms rates.
- 15 Implementation is in SW.
- [0514] After Bit Synch success I_p and Q_p for 20 ms aligned to bit boundaries.
- [0515] If carrier Phase Lock
- [0516] Data Bit = sign (I_p)
- [0517] Else
- 20 [0518] Data Bit = sign[($I_p + j * Q_p$)* exp($j * \text{Carrier Phase Error Estimate}$)]

[0519] Where j is the complex coefficient. This algorithm is a more general case of differential decoding when the carrier phase estimate error is generated from the previous bit.

[0520] Frame Synch. The Frame Synch may operate at 20-10 ms rates and be
5 implementation in SW.

[0521] Bit stream after data demodulation provided as a array of binary values per SV.

[0522] Example approach:

[0523] Cold Start (No Information available)

10 [0524] Preamble Sync. Identify Preamble in the bit pattern. Decode HOW. Identify second preamble 6 seconds off and identify second HOW off by 1.

[0525] Approximate Time Available

[0526] Identify preamble. Decode HOW. If HOW is within time uncertainty declare frame synch complete.

15 [0527] Verify Frame Synch. Repeat the process and confirm HOW change by 1 with 6 second offset.

[0528] Aiding Information Available and 1 SV completed Frame Synch

[0529] The following steps may be executed

[0530] Compute pre positioning for the SVs needing frame synch using time from the
20 first SV frame synch.

[0531] Compute frame starts for the SVs of interest relative to local time.

[0532] Set the frame synch information.

[0533] Aiding Bits available

[0534] Compute a cost function using the aiding bits available.

[0535] $\text{Cost Function}(\text{offset}) = \text{Sum}(\text{Aiding Bits}(\text{bit number}) * \text{Bit Stream}(\text{bit number} + \text{offset}))$

5 [0536] The Sum is carried over all the available aiding bits.

[0537] If SV offset information is available combine the cost function for multiple SVs using the following equation

[0538] $\text{Combined Cost Function}(\text{offset}) += \text{Cost Function}(\text{offset} + \text{SV offset})$

[0539] If $[\text{max}((\text{Combined Cost Function}(\text{offset}) > \text{Threshold}) \&\& (\text{max}(\text{Combined}$

10 $\text{Cost Function}(\text{offset})) - \text{second max}(\text{Combined Cost Function}(\text{offset})) > \text{Threshold}]$

[0540] Declare frame synch Complete.

[0541] The foregoing description of an implementation has been presented for purposes of illustration and description. It is not exhaustive and does not limit the claimed inventions to the precise form disclosed. Modifications and variations are

15 possible in light of the above description or may be acquired from practicing the invention. Note also that the implementation may vary between systems. The claims and their equivalents define the scope of the invention.

PART II
OF
A GPS SYSTEM

5 Field of the Invention

Embodiments of the invention are in the field of systems and methods for processing global positioning system ("GPS") signals, determining geographical location using the processed GPS signals, and related functions.

Background of the Disclosure

10 Systems for processing GPS signals include various hardware and software components generally designed to accomplish the task of geographically locating a GPS receiver as quickly as possible to the highest degree of accuracy possible. GPS systems have become remarkably capable since the very first GPS units became available. Advances in semiconductor technology and microprocessor technology have helped
15 make this improved capability possible. In addition, consumer and government demand for small, fast, GPS-capable devices (for example, E911-compliant cell phones) have helped drive GPS system design improvements.

As used herein GPS capability is the capability to receive and process GPS signals, and further, to use the processed GPS signals to generate accurate position
20 information. More and more devices include GPS capability. Therefore, manufacturers of a wide variety of products desire or require the ability to insert GPS hardware and software into their product designs. Of course, it is desirable for the GPS hardware to use as little semiconductor die space as possible, and for the GPS hardware and software system to be as economical as possible in its demands for memory and power.

25 Designing GPS systems within these constraints usually forces a series of choices among speed, size, power usage, etc. Most existing GPS system designs thus embody a

set of tradeoffs. Most existing GPS systems provide little or no flexibility once designed. For example, they are not readily reconfigurable to process or store data differently under different conditions in order to perform most efficiently at any given time. Typically, the amount of memory made available to a GPS system (either in a stand-alone GPS system,
5 or in another system such as a cell phone) is highly dependent on factors such as the type of signal processing performed and on the absolute limit of memory available.

Another disadvantage of many existing GPS systems is that they rely much of the time on other systems for aiding data in order to provide a position within acceptable time limits. For example, a GPS system in a cell phone may constantly require time
10 aiding, and possibly other aiding, from the cellular network in order to perform effectively. Existing GPS systems may not be capable of acquiring GPS signals within a required time without aiding. This may be acceptable if the time information is always available and if the provision of the time information does not impact other system performance aspects. However, a system that is designed to rely on aiding to meet
15 performance requirements is not capable of performing satisfactorily without aiding, and further is not flexible enough to perform well in both aided and unaided situations.

Brief Description of the Drawings

Figure 1 of Part II is a block diagram of an embodiment of a GPS system.

Figure 1A of Part II is a block diagram of an embodiment of a baseband chip system.

5 **Figure 2 of Part II** is a block diagram showing significant subsystems of one embodiment of a baseband chip.

Figure 3 of Part II is a block diagram illustrating an overview of the general data flow of the GPS system in one embodiment.

10 **Figure 4 of Part II** is a block diagram illustrating general data flow of the GPS system with greater detail.

Figure 5 of Part II is a block diagram showing functional blocks of subsystem 1 in one embodiment.

Figure 6 of Part II is a block diagram showing more detail of subsystem 1 partitioning in one embodiment.

15 **Figure 7 of Part II** is a block diagram showing an embodiment of a subsystem 1 input sample flow.

Figure 8 of Part II is a block diagram showing functional blocks of subsystem 2 in one embodiment.

20 **Figure 9 of Part II** is a block diagram showing an embodiment of a subsystem 2 flow.

Figure 10 of Part II is a block diagram showing another embodiment of a subsystem 2 flow.

Figure 11 of Part II is a block diagram showing an embodiment of a subsystem 2 data flow control.

Figure 12 of Part II is a block diagram showing another embodiment of a subsystem 2 data flow control.

Figure 13 of Part II is a block diagram showing an embodiment of a subsystem 2 control flow.

5 **Figure 14 of Part II** is a block diagram showing an embodiment of a subsystem 2 control module interface.

Figure 15 of Part II is a block diagram of a subsystem 2 master controller flow in an embodiment.

10 **Figure 16 of Part II** is a block diagram of subsystem 3 modules in an embodiment.

Figure 17 of Part II is a block diagram of a subsystem 3 functional flow in an embodiment.

Figure 18 of Part II is another block diagram of a subsystem 3 functional flow in an embodiment.

15 **Figure 19 of Part II** is a block diagram of a subsystem 3 control flow in an embodiment.

Figure 20 of Part II is a block diagram of a subsystem 3 FFT controller flow in an embodiment.

20 **Figure 21 of Part II** is a block diagram of a subsystem 3 FFT flow in an embodiment.

Figure 22 of Part II is a block diagram of a subsystem 3 FFT controller flow in an embodiment.

Figure 23 of Part II is a block diagram of a subsystem 3 master controller flow in an embodiment.

25 **Figure 24 of Part II** is a block diagram of a subsystem 2-FIFO2-subsystem 3 flow in an embodiment.

Figure 25 of Part II is a block diagram of a subsystem 2-FIFO2-subsystem 3-subsystem 5 arrangement in an embodiment.

Figure 26 of Part II is a diagram showing various subsystem 2-FIFO2-subsystem 3 interactions in an embodiment.

5 **Figure 27 of Part II** is a block diagram showing various subsystem 4 modules in one embodiment.

Figure 28 of Part II is a block diagram illustrating aspects of a subsystem 4 sequencer module interface in one embodiment.

10 **Figure 29 of Part II** is a block diagram illustrating a subsystem 4 sequencer flow according to one embodiment.

Figure 30 of Part II is a block diagram illustrating a subsystem 4 sequencer state machine flow according to one embodiment.

Figure 31 of Part II is a block diagram showing various subsystem 5 modules in one embodiment.

15 **Figure 32 of Part II** is a block diagram illustrating aspects of a FIFO 1 structure in one embodiment.

Figure 33 of Part II is a block diagram illustrating a subsystem 5 memory data path flow according to one embodiment.

20 **Figure 34 of Part II** is a block diagram illustrating a subsystem 5 arbitration priority scheme according to one embodiment.

Figure 35 of Part II is a block diagram illustrating aspects of a tracking mode (mode 3) process flow according to an embodiment.

Figure 36 of Part II is a block diagram illustrating aspects of a cold start mode (mode 1) process flow according to an embodiment.

25 **Figure 37 of Part II** is a block diagram illustrating a correlator data path according to an embodiment.

Figure 38 of Part II is a block diagram illustrating aspects of coherent RAM data storage according to an embodiment.

Figure 39 of Part II is a block diagram illustrating aspects FFT access of coherent RAM according to an embodiment.

5 **Figure 40 of Part II** is a block diagram illustrating the order of data needed by the FFT according to an embodiment.

Figure 41 of Part II is a block diagram illustrating a coherent RAM input twiddle select algorithm according to an embodiment.

10 **Figure 42 of Part II** is a block diagram of twiddle generation according to an embodiment.

Figure 43 of Part II is a block diagram illustrating a twiddle multiplexor implementation according to an embodiment.

Figure 44 of Part II is a block diagram illustrating FIFO output address generation according to an embodiment.

15 **Figure 45 of Part II** is a block diagram illustrating aspects of FFT address twiddling according to an embodiment.

Detailed Description

A GPS system and method is described, including at least embodiments of hardware architecture, control logic, signal processing logic, and memory management logic. Embodiments of the GPS system and method are dynamically reconfigurable. For example, memory is dynamically reconfigured to be accessed by various subsystems in different manners under different conditions. Although examples of an overall memory size available to the GPS system and method are described for illustration, other sizes are possible using the same principles described. In addition, access to external memory in addition to the memory shown is possible with the hardware and software described.

The GPS system described processes GPS data at a very high rate, which is an aspect that allows the GPS system to have superior performance without aiding, even in the absence of ideal conditions, such as excellent satellite visibility. In one aspect of the GPS system, extremely powerful time domain matched filtering in combination with fast Fourier transforms ("FFTs") facilitate processing a large amount of data per unit time. FFT operations are used to multiply the number of signal samples used by a factor as high as fifteen or twenty. As an example, the system produces as many as 200,000 effective real time correlators. Conventional GPS systems typically process the GPS signal directly through a number of physical correlators, each of which is actually logic-dedicated to making one correlation. This limits conventional GPS systems to approximately 40,000 effective correlators.

This performance improvement is accomplished by the GPS system with a memory bandwidth consumption of approximately 400 Mbytes/sec. Therefore, the GPS system is capable of producing these correlations with as little as a 64 Kbytes of memory. Compared to the power of conventional systems, the GPS system is more efficient in terms of power usage, cycle-count, and memory usage. As described more fully below, this is partly made possible by the optimization and configurability of multiple subsystems.

The GPS system described is capable of operating in various modes of GPS signal acquisition and processing, as most appropriate for a situation. The GPS system switches between modes automatically as directed by a host processor. The various subsystems may be configured differently in different operations modes. The operational modes will
5 be described in more detail later and include: mode 1, a "cold start"; mode 2, a "coarse-aided acquisition" mode; and mode 3, "high-resolution", or "tracking" mode.

Figure 1 of Part II is a block diagram of an embodiment of a GPS system, including radio frequency ("RF") components and a baseband components. In one embodiment, the RF components and the baseband components interface to an original
10 equipment manufacturer ("OEM"), or "host" processor and OEM memory through an OEM bus. As will be described below, the baseband components include memory components. Optionally, the OEM memory is not required to be accessed by the baseband components. Other possible arrangements include all of the RF components and the baseband components on one chip with all of the required memory and
15 processing power to perform GPS functions. The GPS system is capable of operating effectively without aiding information, or alternatively, it may operate with aiding information from a variety of sources.

Figure 1A of Part II is a block diagram of one embodiment of a baseband chip, including a digital signal processor ("DSP"), an ARM processor, clock components,
20 various memory components, various interface components for external and internal communication, etc.

Figure 2 of Part II is a block diagram showing significant subsystems of one embodiment of a baseband chip, including an input sample subsystem, a signal processor subsystem, an FFT subsystem, a memory subsystem, a sequencer subsystem, and another
25 "miscellaneous" subsystem. For convenience herein the subsystems will be referred to as follows:

subsystem 1 = input subsystem;

subsystem 2 = signal processing subsystem;

subsystem 3 = FFT subsystem;

subsystem 4 = sequencer/control subsystem; and

subsystem 5 = memory/arbitration subsystem.

The division of tasks or functionality between the subsystems is a design choice.

5 In different embodiments of the invention, the different subsystems may share functionalities in different ways, or there may be more a less subsystems. For example, in some embodiments shown herein the sequencer/control subsystem is not a separate subsystem. Rather part of the sequencer functionality resides on subsystem 2 and the remaining functionality resides on subsystem 3.

10 **Figure 3 of Part II** is a block diagram illustrating an overview of the general data flow of the GPS system in one embodiment, and further illustrating a conceptual arrangement of various subsystems. As shown, subsystems 1, 2, and 3 arbitrate for access to a random access memory ("RAM"). As described further below, the RAM is logically divided into four sections, but in various embodiments can be any number of
15 physical memory devices, either internal to the GPS system, or external to the GPS system.

A signal flow for one system embodiment is shown in **Figure 4 of Part II**. An RF signal is received in the input sample processing block or subsystem 1. Subsystem 1 includes a counter chain that is a divide-down from an input sampling clock and other
20 clocks that provide time and frequency references against the input sampling.

Figure 5 of Part II shows basic functional modules within subsystem 1.

Figure 6 of Part II shows additional detail of the basic functional modules within subsystem 1. The automatic gain control ("AGC") looks at the RF samples coming into the input sample block. There is a down-counter in the AGC. Input samples have a
25 magnitude of one or zero. A value of one corresponds to a strong signal. When a sample has a value of one, the AGC counts down. If it counts down too much, the signal is too large, so AGC send control signal to RF telling it to decrease the gain, and vice versa.

Figure 7 of Part II shows additional detail of the basic subsystem 1 data flow.

Figure 8 of Part II shows different blocks in subsystem 2, the signal processor subsystem.

Figure 9 of Part II shows of a data flow for subsystem. In various embodiments, the cross-correlation function is omitted. In other embodiments, the data wipe-off and collect function is performed in subsystem 3.

Figure 10 of Part II is an alternative block diagram of subsystem 3 without the cross-correlation elements shown in **Figure 9 of Part II**.

Figure 11 of Part II shows a data flow for an embodiment of subsystem 2. This diagram illustrates pipeline stages and registers for the flow control. Each titled section, such as "memory interface", "signal processor", etc., has a control mechanism and each acts as its own state machine. There are hand-offs between the sections as data flows left to right.

Figure 12 of Part II is another diagram of data flow in an embodiment of subsystem 2 that shows critical paths to be considered in timing analysis.

Figure 13 of Part II is a diagram showing a subsystem 2 control flow in one embodiment. A master control is to the left of the diagram. The matched filter controller is programmed to process for a number of milliseconds, and the code and carrier NCO are programmed with starting positions. The matched filter controller draws data from the FIFO1 to the code and carrier NCO and then into the signal processor and from signal processor to the matched filter, then to the coherent accumulator and then to coherent RAM, which coherently accumulates.

In one embodiment, there are different ways to implement the matched filter so that it processes data for a certain number of milliseconds. The matched filter draws data from the signal processor until it stops requiring data. Then the signal processor stops producing data. The coherent accumulator takes whatever the matched filter outputs and coherently accumulates it. The matched filter decides when to change channels to

process data from a different channel, or data from a different satellite vehicle. The matched filter effectively says “stop processing” when it has processed for the period of time.

5 **Figure 14 of Part II** Is a block diagram of an embodiment of the controller on the left of **Figure 13 of Part II** showing more detail regarding its communication with different modules (e.g., signal processor, matched filter, coder, etc.). The controller initializes processing. There is a channel RAM region of RAM accesses by the subsystem 2 control module. This is part of an embodiment in which the sequencer functionality is divided between subsystem 2 and subsystem 3. The portion of the
10 sequencer instantiated in subsystem 2 is shown here. When a channel is initiated, the control module accesses the channel RAM and pulls in channel parameters required for subsystem 2 to process that particular channel. It then programs the signal processor, the coder, the matched filter, and the coherent accumulator with their parameters required to process that channel. The parameters, for example, tell the matched filter to run for a
15 certain number of milliseconds, then when it is done the matched filter tells the subsystem 2 control module know that it is finished, and the controller can move on to the next channel. There is a linked list in the channel RAM that stores the location of the channel RAM for the next channel, and so on.

20 In one embodiment, different channels are programmed to operate in different modes as enumerated above (cold start, coarse-aided acquisition, and tracking modes). The parameters stored in the channel RAM allow the subsystems to be programmed appropriately for a particular channel in a particular mode. The channels access the subsystems in a time multiplexed manner.

25 **Figure 15 of Part II** shows a master controller flow for one embodiment of subsystem 2.

Figure 16 of Part II shows various modules and module partitioning of one embodiment of a fast Fourier transform (“FFT”) subsystem 3.

Figure 17 of Part II shows more detail of an embodiment of subsystem 3. A subsystem 3 control module is shown at the bottom of the diagram, and is similar in function to the subsystem 2 control module. The dashed lines represent the interface from the subsystem to memory. Peak RAM, NCS RAM, etc, are regions in memory.

5 The coherent RAM is also a region in memory. The FIFO2, however, is a FIFO2 control structure that controls the access between subsystem 2 and subsystem 3 to the memory. For example, the FIFO2 lets subsystem 3 know when there is data to operate on in the coherent RAM. FIFO2 also lets subsystem 2 know when it is about to overwrite data that subsystem 3 has not used yet. There is a corresponding FIFO1 between subsystem 1
10 and subsystem 2 that control access to the input sample RAM.

Figure 18 of Part II shows a subsystem 3 data flow, including FIFO2. The data going from subsystem2 to subsystem3 is controlled by FIFO2 structure. The actual data is in the coherent RAM, right above FIFO2. The FFT takes a certain number of sample inputs out of the coherent RAM and performs an FFT on them. The FFT generate a
15 number of frequencies as dictated by the particular processing mode. As and example, for an 8 sample, 16 point FFT, the FFT generates 16 frequencies. Not all of the 16 frequencies may be of interest. For example, the outer frequencies may not be useful... The gather module accepts only a programmed number of frequencies. The number of frequencies is programmable.

20 The gather module selects the desired frequencies, packs them in a more compact form and passes them on to the NCS, which is a non-coherent accumulator. Although not shown here, the output of the gather mod goes to the track history RAM ("TH"). The output of the gather module is coherent data (still having distinct I and Q components). Non-coherent data is stored in NCS. Selected coherent data is stored by the TH for later
25 examination and processing by software. At the top of the diagram, there is a line that goes to the sorter threshold module. There is a large amount of information generated by the NCS module. It may not be desirable to save it, e.g., in an initial acquisition mode. It may be desirable to process, for a particular satellite vehicle, odd and even half-chip and multiple frequencies, each of which will generate a relatively large data array. So as the

data is generated, the data is processed and peak sorted. The eight largest values are stored, as well as information regarding where they occurred, such as what tap number, what frequency, and whether it was at an odd or even half-chip. At the end of a context that is all stored in the peak sorter.

5 Context is a term used herein to denote the completion of processing of one channel for a subsystem. At the end of the context the peaks are saved, either for when the channel is about to enter subsystem 3, or when processing is almost done, so that software can examine the peaks and make its determinations.

10 **Figure 19 of Part II** shows timing signals for a subsystem 3 control flow. Two timing division terms are used herein, e.g. T1 and PDI (or pre-detect integration) will now be defined. A T1 is basically how long it takes to coherently accumulate in the coherent accumulator which is fed from the output of the matched filter.

15 Suppose the matched filter is operating in a full matched filter mode. The output of the matched filter is a full millisecond worth of coherent accumulations. The coherent accumulations take the full millisecond allotted for coherent accumulations. Each time processing includes a particular tap position, there will be another full millisecond of coherent accumulations produced. The coherent accumulator adds a first and second millisecond worth of coherent accumulations to get 2 milliseconds of coherent accumulations. More or less than 2 milliseconds of coherent accumulations can be
20 programmed. The completion of the programmed number of milliseconds of coherent accumulations for a tap constitute a T1's worth of coherent accumulations. There are variations with different modes, such as in the length of T1 in milliseconds. In some modes, it is desirable to accumulate longer to look for a weaker signal. When coherent accumulation is finished, non-coherent accumulation starts

25 For a 1/8 matched filter (a low-resolution mode), then each one of the 1/8 fractional blocks accumulate 1/8 ms in one shot. The matched filter is thus run through 8 times into the coherent accumulator to produce one ms worth of coherent accumulations.

This is repeated as required to get a predetermined number of ms of coherent accumulations.

Each T1 represents one input sample into the FFT. For example, for an 8 sample, 16 point FFT, 8 T1s are required to feed into the FFT. This is called a PDI; 8 T1s in one
5 shot constitute a PDI.

Each T1 represents an input sample to the FFT. However many T1s are fed into the FFT at one time constitute a PDI.

Figure 19 of Part II illustrate the need to keep track of T1 boundaries and PDI boundaries. This is facilitates by the "done" signals. **Figure 19 of Part II** shows a
10 pipeline flow starting from the FFT to the gather module to the NCS. The subsystem3 backend controller initiates the subsystem. The pdiNext signal starts the FFT processing data. Then the NCS needs to let the backend controller know "I completed processing that PDI, what next"?

Figure 20 of Part II shows detail of the FFT controller, including time and
15 address counters. The FFT controller keeps track of the location of data in the coherent RAM that is required to feed the FFT. The controller that knows what cycle the FFT is in and what information the FFT needs. The controller generates an address and an indication of the status of FFT processing.

The FFT core includes a four point FFT. In order to configure a 16 point FFT the
20 4 point FFT is run repeatedly in a particular pattern to generate the 16 point FFT. Since this core is a 4 point FFT it needs 4 T1 input samples at a time. However, subsystem 2 generates the T1s in a linear fashion. The FFT requires the T1s them in a bit reversed order. For example, say the FFT needs 4 T1s at a time out of 8, being 2, 7, 3, and 1 on the first shot and some other pattern on the next shot. A twiddle algorithm stores the
25 output of subsystem 2 into 4 diff physical RAMS in an optimal pattern such that the data required by subsystem 3 can be accesses 4 T1s at a time without a data collision. Without the twiddle algorithm and dividing the RAM into four areas, it would be necessary to perform more memory reads. The algorithm knows which 4 T1s the FFT

will need at one time and assures that they are stored in different RAMs (or RAM areas). **Figures 37-45 of Part II** illustrate hardware and logic aspects of FFT address twiddling in an embodiment.

Figure 21 of Part II shows a basic subsystem 3 flow in one embodiment. The scaling module shown performs descaling. The descaling is necessary because the data is scaled as follows. In the coherent accumulator, there is a relatively narrow pipeline for the data to flow through, and the dynamic range is very large. Given, for example, only 8 bits of I data and 8 bits of Q data for which there is room in the coherent RAM. The 8 bits of I and Q are thus associated with an exponent, or scale. There is an auto function such that bumps the exponent by one and scales back the 8 bits by one when the accumulator nears overflow for the 8 bits. Scaling is done by subsystem 2 when it feeds the data into the coherent RAM. Each T1 has an assoc scale values. Different scale values cannot be fed into the FFT, such as when 4 T1s at a time are fed into the FFT. So that is what the scale module normalizes all the data to the same scale. The output of the FFT goes to the NCS, which has its own scaling function. Coherent and non-coherent scaling takes place.

Figure 22 of Part II shows how the FFT cycles through under control of the subsystem 3 controller. The FFT processes for a particular chip position, completes the FFT and then needs to be told to advance to the next chip position. This is a detail of the FFT controller state machine.

Figure 23 of Part II shows a flow chart for the subsystem 3 master controller. A semaphore word indicates various data that must be quickly disseminated between the subsystems on start up. The data is required to be exchanged between three quasi-asynchronous processes: subsystem 2, subsystem 3 and software. Each one runs on its own and gets information from the others. A lot of the handshake information required between them is in the semaphore word. Information in the semaphore word includes, is a 100ms report going to be created, is a phase adjust in subsystem 3 to be performed, various status flags from one subsystem to another, etc. The controller reads the semaphore word and makes decisions. The controller writes back to the semaphore

word, for example, saying subsystem 2 is now processing a particular channel. Software can read that semaphore word for the channel, for example, when software sees an active bit on, it knows the channel is being currently processed by subsystem 3.

5 The controller initializes subsystem 3 and FIFO2 (see near top center) that basically tells it to go out to the channel RAM, pull out the channel parameters and program subsystem 3 for the channel. The "FIFO2 data available" says subsystem 3 needs data from the coherent RAM, so it needs to know if the data is available. If it is not, it waits for it to become available. This involves lapping rules which control a lot of the sequencing between subsystem 2 and subsystem 3. Subsystem 2 and subsystem 3
10 process a linked list of channels. Subsystem 2 processes a channel and moves on, and then subsystem 3 to come into the channel, takes the data out and processes it. Subsystem 2 should not get ahead of subsystem 3. Subsystem 2 can catch up to subsystem 3 but must be just behind it. This is controlled by the lapping rules, explained in more detail below.

15 The "FIFO2 data available?" decision block and the "channel in subsystem 2?" decision block are part of the lapping rule. If data is not avail, subsystem 3 is directed not to leave the channel yet, because subsystem 2 may be producing data for subsystem 3. When subsystem 2 leaves, subsystem 3 may leave the channel.

20 Cross-correlation functionality is also shown. In one embodiment, subsystem 2 produces the basic cross-correlation data, and subsystem 3 actually removes the cross-correlation data.

Referring to the "FFTT1 done?" decision block, there is a bit synch mode in processing is performed for multiple T1 offsets within a PDI. If not in bit synch mode, every time, e.g., if it is an 8 sample, 16 point FFT (this would not be a bit synch mode),
25 after 8 T1s are processed into the FFT, the next full 8 T1s, or the next PDI, is fed into the FFT. On the other hand, in bit synch mode, there are 20 T1s because each one is 1 ms. It is thus necessary to figure out 20 hypothesis bit positions. There is a 20ms ambiguity in the bit positions. There are 20ms in a GPS data bit, and the CA code repeats each ms.

An alignment can be performed in the CA code. With bit synch, the one ms ambiguity within the 20 ms is removed. Different T1 offsets are tried for different hypotheses.

When a PDI is completed, the flow goes to the bottom right of the diagram. It is determined whether another odd or even ,or another frequency is desired. Subsystem 2
5 generates even and odd multiple frequencies that subsystem 3 needs to process. A HW tracking loop is iterated, certain scaled and bias parameters are updated as shown on the right upward-going flow. On the leftmost flow is the cleanup for the channel when the context is about to end. The semaphore word is updated to reflect whatever happened during the last context for the channel. HW tracking, and report generation (100ms
10 report or context report as appropriate) also occur. Then the channel is deactivated, and the active bit in the semaphore word is set to zero. The linked list in the channel RAM is consulted to determine which channel will have access to the hardware next.

Figure 24 of Part II shows interaction between subsystem 2 and subsystem 3 through the FIFO2. Subsystem 5 is the memory arbiter. As discussed, the RAM is
15 divided into the 4 memory blocks shown. Only one of the subsystems or one functionality in a subsystem can access the RAM at one time. This shows subsystem 2 producing coherent data and trying to write it to the Ram and subsystem 3 requesting data for processing. Subsystem 5 decides who gets access. The FIFO2 lets each subsystem know the status of their respective requests to access RAM.

20 **Figure 25 of Part II** is a diagram showing another detail of subsystem 5 functionality.

Figure 26 of Part II illustrates an embodiment of the lapping rules. If a subsystem is between the dashed lines, it is operating. Circle1: a channel is getting into subsystem 2. circle2: a channel enters subsystem 3 for the same channel. As the note
25 says, subsystem 3 may not leave until subsystem 2 leaves. Subsystem 3 cannot lap subsystem 2 in other words. circle 3; subsystem 2 leaves the channel, so subsystem 3 may now leave. Circle4: subsystem 3 still may be processing, subsystem 2 laps all the way around and catches up. Subsystem 2 may not enter until subsystem 3 is done and

leaves. Circle5: subsystem 2 can now enter. Subsystems 2 and 3 can be in the channel at the same time, but only if subsystem 2 entered first.

Figure 27 of Part II shows elements of the sequencer subsystem implemented in subsystem 3 in one embodiment.

5 **Figure 28 of Part II** is a block diagram of a sequencing module that is one implementation of the **Figure 23 of Part II** flow chart. The sequencer module includes control elements and also includes registers, data storage, interface to memory etc. However, the sequencer module interface shown is basically flow control. The arithmetic and logic unit (“ALU”) module provides specialized arithmetic capability for the HW
10 tracking functionality. The backend controller knows where the NCS module is, and it controls the NCS module in a similar fashion to the FFT controller controlling of the FFT.

In addition, there is a HW tracker that has functionality that occurs on a PDI rate and functionality that occurs on a context rate (that is, before processing of a channel is
15 complete). The sequencer controls the HW tracking loop functionality using the ALU as its number cruncher.

Figure 29 of Part II is a block diagram of an embodiment in which the sequencer functionality is contained in a single module. This is an alternative embodiment to that previously described in which the sequencer functionality is shared between subsystem 2
20 and subsystem 3.

Figure 30 of Part II is a block diagram illustrating some of the functionality that the sequencer implements. The functionality includes data path control, hardware tracking loops, and report control. Report control controls the generation of various reports, such as a 100ms report, a context report, a peak report, etc., for the use of
25 software.

Figure 31 of Part II illustrates various RAM functionalities. In one embodiment, all of the RAM functionalities are included in one physical RAM. In other embodiments,

the RAM functionalities are shared between different physical RAMs. A memory subsystem handles arbitration of requests for access to these RAM functionalities.

5 **Figure 32 of Part II** illustrates details of an implementation of the FIFO1. The line, block and wrap signals on the left are used by subsystem 2 when it is trying to write data into the RAM. On the right, the line, block and wrap signals are used by subsystem 3 when it is trying to read data out of the RAM.

10 **Figure 33 of Part II** is a block diagram of a memory data path flow that shows different types RAM and how different subsystems request to access it. Subsystem 1 is writing data into the input sample RAM (on the left), and subsystem 2 is reading data out under control of FIFO1. Similarly, FIFO2 controls access to the coherent RAM by subsystem 2 and subsystem 3 is subsystem 3. The TH RAM stores track history in the form of coherent data, as previously described. The bit synch RAM, in one embodiment, is a type of TH RAM for a particular channel in which there are multiple T1 offsets. Software uses the multiple T1 offsets to determine where a data bit edge is. During system operation, the data subsystems are trying to access these different RAMs. The sequencer is trying to access the channel RAM primarily, but it also tries to access the coherent RAM, for example to access pointers. Software also tries to access RAM. Notice the arrow from the software to the channel RAM. Software examines the NCS RAM, TH RAM and peak RAM. All of the RAM access is handled by subsystem 5 arbitration.

20 **Figure 34 of Part II** is a block diagram showing subsystem 5 and an example of arbitration priorities. One possible assignment of priorities is listed on the bottom left. Typically, subsystem 1 has very high priority because it is receiving data from satellite vehicles at an uncontrollable rate. A central processing unit ("CPU") typically has a high priority as well.

25 **Figure 35 of Part II** shows some of the processes that take place during system operation, particularly in mode 3, or tracking mode. For example, on the left of the diagram, there is a timing chain, and then the input storage that goes into the input RAM.

Data should be written to particular locations in RAM. In one embodiment, there is a free-running, 32-bit counter in a timer module of subsystem 1. The counter is associated with user time. User time indicates a rate at which the samples coming from the RF are strobed. User time is associated with where data is stored in the input sample RAM.

5 Subsystem 3 understands that the location of data in the input sample RAM is associated with time. Subsystem 3 pulls that data out into the matched filter. A signal processor (not shown) in front of the matched filter does decimation and rotation of the signal for the matched filter.

10 At the top left there is a dashed line showing user time to the left and process time to the right. Conceptually processes to the left are tightly related to the RF clock. Events must happen on user clock periods. To the right there is a signal processor clock that may have any practical frequency. The higher the signal processor clock frequency, the more data is processed per unit time.

15 **Figure 36 of Part II** shows some of the processes that take place during system operation, particularly in mode 1, or cold start mode. In mode 1, the coherent RAM is not shown in the path between subsystem 2 and subsystem 3. In mode 1, data goes straight from the matched filter into the FFT. The matched filter is said to be in a "locked" mode because matched filter and FFT are locked together. In this mode, the matched filter is in a fractional mode, and is divided into four sections.

20 In mode 1 a fractional matched filter is used differently than in other modes. For example, the matched filter is divided into four sections, each of which is a separate input into a 4-sample, 8-point FFT. So the FFT gets four inputs directly from the matched filter. Each of the points, or samples, is $\frac{1}{4}$ sec worth of coherent integrations. In an FFT, a shorter coherent integration time for each sample results in wider frequency or bin
25 widths. If coherent accumulation occurs for longer period (meaning there are longer T1 periods), the frequency bins are narrower. Narrow frequency bins are desirable for tracking, but not desirable for a wide search.

General System Operation

In subsystem 1 there is a timing module, a decimation module that massages the signal from the RF, and an automatic gain control ("AGC") module that controls the RF signal coming in.

5 The decimated signal then goes through a FIFO-type control and is stored in a section of RAM. There are three types of RAM: input RAM; coherent RAM; and non-coherent RAM. These three types of RAM are physically implemented in one RAM that is dynamically allocated by regions. In one embodiment, the RAM is divided into four blocks, for purposes of the FFT and feeding of data to the FFT. A twiddle algorithm
10 allows data to be stored in such a way that four pieces of data come out of the RAM at one time. A four-block RAM helps avoid data conflicts and facilitates throughput access to the RAMs.

The three data subsystems 1, 2, and 3 all request access to the four RAMs, and the different regions of the RAMs storing an input sample, a coherent accumulation, and a non-coherent accumulation.

15 When subsystem 1 is initiated, it stores decimated data into the input sample RAM. In one embodiment, there are two modes of storage. One mode is a one-shot mode in which there is a region of address space that is filled up in one shot. The rest of the subsystems then operate on the data. Another mode is a cyclic buffer mode that also fills a particular region. When the region is filled up, the write operation cycles back to
20 beginning of the region and begins overwriting the data stored. For the cyclic buffer mode an unspecified number of channels time multiplex the use of subsystem 2 and subsystem 3 and they each cycle through and get their respective opportunities to process the data.

25 The operation of the system is fundamentally different in the three operational modes as will be explained further below. Some basic commonalities apply to all of the three operational modes, however. In general for example, input samples are being saved, correlations are being performed, and the spread spectrum is being moved,

despreading for the matched filter ("matched filter"), and the carrier phase is being moved in the signal processor.

In one embodiment, length of the matched filter is 1024, which is significantly longer than a conventional matched filter, which may have a length of 11. The FFT
5 performs as many as 32 point transforms and can be used any time for various different array size and shapes.

In a case where there is huge uncertainty, (e.g., not all satellites in view, etc), the spread spectrum processing may require a search of 1024 chip offsets in the time domain. A replica code is lined up with the receive signal in order to get a match and maximum
10 output; when it is misaligned, there is reduced output. There will be a signal power peak when the correlation is aligned correctly. The matched filter keeps a copy of the special replica code in a register and slides the signal past it; and about once every ms a peak comes out. In an outdoor environment, a single pulse can almost be detected when there is a strong signal, but usually many must be averaged together. Many of those are
15 summed, and for each offset there is an addition, a skip ahead one ms, and another addition. 1024 additions may be saved for each code offset.

In one embodiment, the useful window on the correlation when integrating for a full ms of code, is about + or - 500Hz (with some loss). Usually, integrations are placed about twice that close. Considering an oscillator that is not extremely accurate, e.g., that
20 might have tens of KHz of error, the frequency is adjusted and the correlations repeated, perhaps as many as 200 times, in the frequency domain. So just to search one code when the oscillator is not calibrated takes perhaps like 200K individual search bins, or 400K because when the code is spaced at two sample per chip. Now we have 200 frequencies multiplied by 2000 correlations, or 400K individual cells to be searched. These must be
25 averaged together multiple times. This case is one example scenario of matched filter function, but is usually reserved for an outdoor environment where a strong signal can be expected.

General Memory and Arbitration Aspects

FIFO1 is an address generator. In the figures, FIFO1 is shown with multiple conceptual inputs, but in one embodiment the RAM that includes FIFO1 has one port. All the inputs to the RAM go through the arbiter. FIFO1 counts an address in a circular pattern to the right. FIFO1 is directed to start at an address and make a buffer of a certain length. The counter counts from the start address to the end and goes back to the start and begins overwriting the data. On the read side there is a similar address counter that gets reloaded for each signal or logical channel that is being created. So the FIFO is reloaded, some samples are played, and the FIFO is reloaded again more samples are played. Basically with a 50MHz clock it is possible to replay a sample about 25 times in real time before it is used up, or overwritten. To keep up with the flow, approximately 24 channels can be allocated.

FIFO1 counts addresses; writes go in at a relatively slow rate, perhaps every 128 clock cycles or so, and another sample is dumped. FIFO1 is accessed in 64-bit lines. About every eighth microsecond another line of samples is collected and FIFO1 asks for a cycle of memory. FIFO1 goes through the arbiter, and because it is high priority, it usually gets the cycle of memory requested. There is only one higher priority – the microcontroller. FIFO1 writes the line and go on to collect another 32 samples.

Subsystem 2 does much the same thing. The sequencer puts the address of the sample it wants into the FIFO. Basically, the delay of each satellite being searched dictates the time at which the first sample was collected. There is a correlation between a counter that counts the samples coming in and the phase the system wants to start at. Logic in the FIFO turns the sample count into a memory address. The FIFO asks for a cycle to read and gets a line of 32 samples. This is repeated and the FIFO feeds the data into the matched filter.

Referring to **Figure 4 of Part II**, data comes into the input subsystem 1 at a relatively slow rate, a counter counts an address and a similar counter in subsystem 2 is counting an address in a loop. The input sample buffer is one is reloaded 25 times, and the processing is timed so that the samples required to be processed are the earliest in time. After processing, some samples have already disappeared from the buffer and

some new ones have been stored. The processing is timed to take about the same time it takes to make a first pass (e.g., 800 micro sec). The samples are all spread out in time and the buffer is constantly reloaded. The summation process reads the previous data out of RAM adds the new data to it and puts it back in RAM; this is a coherent accumulator
5 block. For each sample produced it adds to the memory, and once the whole PDI is produced, the FFT block takes the first chip offset for each of these, processes it, take the magnitude (a real positive number) and add it to a bin for that offset. The PDI might be 7ms, it might be 7 samples/each code offset, so memory will be arranged as 1024 for the first ms, 1024 for the second ms, etc.

10 The sequencer ports read memory as well. So there is a constant data pattern and various subsystems are making their requests for memory cycles. The arbiter takes the requests from the subsystems, decides which has highest priority, and grants memory cycles. In one embodiment, the priority is hard-wired, as shown in **Figure 34 of Part II**. The data path is switched to allow the appropriate data port to access the memory and the
15 memory bus is switched. Then the arbiter moves on to next highest priority. In one embodiment, 6-8 ports being switched according to their priorities.

General Sequencing Aspects

The sequencer, in on embodiment, is a loader including a portion of memory is allocated with a state of each of the channels. The sequencer's job, simply stated, is to
20 take the data in memory, and run it in the hardware as long as it is supposed to run. Everything the sequencer needs to know is in approximately 256 bytes of memory. The sequencer traverses a list and loads and restores state as operations occur in the various subsystems. The sequencer must keep track of what data is being processed in which subsystem. Typically, the RAM has some addresses reserved for state. In general,
25 however, here are no limitations on where to store data, or how much data to store, except data cannot overlap.

Periodically the state of a channel is saved as a report that enables recovery of the exact state of the channel. The reports contain the precise information about where the

channel was. The reports are typically accurate to within nanoseconds or even picoseconds (in the case of carrier) to extract the coded carrier phase. This is a big, high resolution number with an angle and a delay for each of the samples. The report is dumped out periodically, e.g., every 20ms, which makes it possible to trace through what the hardware is actually doing. In concept, this is like a direct memory access ("DMA") engine, but includes logic to perform additional functions under external direction, for example, from a host processor.

The processor directs the sequencer to load the state of the system, including all the modes, and the carrier code NCO phase from where the system left off last time, and to process so many samples. The sequencer is then directed to move on to the next channel. Various data are generated by the sequencer, including error conditions, etc. There is a whole sequence of actions such as checking the channel is on, checking to see if any statuses are wrong (for example, such as would prevent the system from processing), trying to load the phase in and compares the phase to the state of the input FIFO. If the sample the sequencer wants is not in the buffer, it does not try to run. It looks at a margin from one sample to another, and determines whether margin is above a minimum. If the margin is not above the minimum, then it does not run. The sequencer does all that and then moves to the next channel or process. Sometimes the buffer is already overflowed and there is no chance of recovery, in which case the sequencer will shut that channel down without wasting further energy figuring out it is hopelessly lost. The sequencer sends an interrupt to the host processor so that problem can be fixed before the user notices.

The sequencing is dependent on how many ms are processed by particular subsystems, which is in turn based on data availability. Part of the information loaded by the sequencer is whether there are enough samples in a buffer to run a channel. Sometimes the sequencer is pointing at a shared buffer, sometimes at a dedicated buffer. The sequencer looks at how many samples it needs to make a particular kind of FFT vs. how many samples are in the buffer. The sequencer looks at whether the sequencer was requested to wait until there are enough samples in the buffer. For example, the

processor may still be filling the buffer, so the system either needs to wait for the buffer to fill or just get out. If the sequencer waits, comes back and there are still not enough samples, it just skips. This kind of logic is going on in the sequencer: sorting data (what data goes with what channel), reloading all the addresses of the buffers; checking the
5 states of the buffers; running when the system can run, skipping when the system cannot run; and staying in synch with the others subsystems. There are scenarios in which one subsystem is running channels ahead and another subsystem is actually processing delayed data because he has a heavy workload at some point in time. Later the other subsystem may have no work load and will catch up. Because the processing of one
10 subsystem is more bursty, while the processing of another subsystem is more regular, the lagging subsystem can usually catch up. In other scenarios, a buffer is full and a subsystem seeking to put more data in the buffer must to stop. The subsystem may have to stop for such a long period that the next data it needs is lost already. This also generates an error condition which must be responded to.

15 There is messaging between the sequencer and the processor. In general, the processor is watching what is going on continuously by the status that is being dumped out and the data that is coming out.

The GPS system further includes a track history element that provides advantages in controlling the system based on historical performance. The track history element
20 ("TH") is shown, for example, in **Figures 4, 17, and 18 of Part II**. The TH data can be analyzed and used in many ways, such as calculating the carrier phase error and sending a correction. This is useful because as satellite move or the oscillator moves, adjustment, for example, to frequency, should be made.

In one embodiment, the TH is a buffer of coherent integration samples. The
25 coherent integration samples are relatively long. The length of the integration samples, however, is limited. For long summations the carrier has to be perfect for long periods of time, which is virtually impossible. The length of the integrations can be as much as about 20ms, which is the size of one of the data bits on the modulation.

One of the uses of the TH is keeping a small window of about, 4 chips (where there are 4 samples/chip, or 16 samples every 20ms) for observation. The TH may begin overwriting perhaps every 200ms. The history in the window reveals what happened to the carrier phase and what happened to the data. The TH data can be "post processed" in more sophisticated ways than the real time data to provide useful information about system performance in the recent past.

Various Operational Modes

As previously described, the various subsystems may be configured differently in different operations modes. To reiterate, the operational modes include: mode 1, a "cold start" or "locked" mode; mode 2, a "coarse-aided acquisition" mode; and mode 3, "high-resolution" or "tracking" mode.

"Cold Start" Mode 1

Mode 1 is typically used when the GPS system is complexly "lost". For example, the GPS system will be lost if it has no information regarding time that could be used to predict where the satellites are, or it does not know its physical location well enough to have an idea where the satellites are. Mode 1 is thus a blind search. This involves going down entire list of satellites, processing 400K offsets one satellite at a time. There are 32 satellite codes. Sometimes the GPS system will know which 24 are possible (based on the portion of the sky that is visible), and sometimes it will not. The blind search involves millions of individual correlations times hundreds. In mode 1 it is not possible to save all the individual bins over time. A little slice of that data can be processed, however. In mode 1, a one-shot capture of the signal is saved in the input RAM, including perhaps as much as 100ms of signal. A compressed mode (x4 compression) I used, so there is 100K bytes of data n 100ms of signal. There is one 8K buffer on the output, and in that buffer this 100ms signal is replayed. Effectively 1K correlations times 8 frequencies (FFT makes 8 frequencies for each of the 1K chip offsets and saves that in the RAM) are performed. The data in the buffer is processed and only strong signal peaks are stored. Then the process is repeated with a half chip offset. In mode 1, there

are many bins of output, and a long sample helps find the weakest signal that needed to acquire. In one embodiment, mode 1 is limited to about 30DbHz, which is standard threshold for outdoor environment. The search is continued until something is found. The signal eventually found is used to set time and try to improve uncertainty for
5 subsequent searches. In the initial search process about 25K correlations times eight, or about 200K correlations are performed. That is to say, effectively, about 200K one ms correlations per second are being performed. We will refer to this as 200K effective correlators.

Referring momentarily to **Figure 3 of Part II**, in mode 1, FIFO1 is configured to
10 be relatively large. In mode 1, samples are fed directly into the FFT, as shown by the dashed line between subsystem 2 and subsystem 3. The FFT is essentially summing magnitudes into the output RAM and produces the 8 frequencies without going through FIFO2.

"Coarse-Aided Acquisition" Mode 2

15 A next mode, Coarse-Aided Acquisition mode, or mode 2, is a more sensitive case and can be used in a wide variety of scenarios. For example, if a user carries the GPS system around and enters a basement or underground garage, the GPS unit uses the previously known or last known data.

Mode 2 uses a modest size input sample, such as 10ms or less. The input buffer is
20 continuously filled (rather used as a snapshot buffer) and set up a single shared coherent buffer that is on the order of 15-20Kbytes, or enough to do one FFT. For example, the system might want to save seven ms of 1024 correlations in the buffer (2 bytes apiece) so the buffer would be about 14K. The 14KB of samples are taken as individual code phases and passed into the FFT. There are seven time-delayed samples at a time going
25 into the FFT. A sixteen point transform is produces and several of the outputs, maybe seven or nine, are kept in a center frequency bin. In mode 2 there are multiple copies of the output buffer which fill most of the RAM. Remaining RAM is used for dedicated non-coherent summation buffers. For example, given 128K of RAM, 100K individual

bins are possible in big arrays, each of the bins being a frequency or code offset. In mode 2 we are able to search only 100K correlators at one time given the example of 128K of memory, or about 40K correlator limited by 64K of memory. This means 40K or 100K correlator, effective real time, continuous processing. The result is a stream of seven ms samples sitting in a buffer. The stream is processed for even and odd half-chip offsets. There is a one-chip space in the correlator and the correlator can be replayed twice (in a full matched filter mode) to get a half chip offset. The number of frequencies is programmable. For example, perhaps seven or nine frequencies are generated. This results in relatively large arrays of nine frequencies by 1K chips and another array with a half chip shift. Moving to a new frequency or new satellite, the process is repeated until the memory is full.

In mode 2 we location is known at least to the accuracy of a continent, and the time is known to at least wrist watch accuracy. Usually some aiding information is required, because the signal strength is too weak to recover data sequence off the satellite. Aiding information may include a satellite list, or the locations of satellite with respect to each other. If the oscillator used in a high quality oscillator, the requirement for aiding goes down. Of course, the uncertainty goes down when one or more pieces of information are available to certain accuracies.

In mode 2 the GPS system is configured to have relatively a small input, a coherent buffer is present, as much memory as possible is reserved for output, and processing occurs continuously. Switching between GPS signals, one of the buffers is filled, the whole list is processed, and by the time the sequencer comes back to the buffer seven ms of new samples are in the buffer. The new seven ms are added on top of the old seven ms, and this continues as long as the weak signal can be detected.

"High-Resolution" or "Tracking" Mode 3

A third mode, a "high-resolution" mode, or "tracking" mode will now be described. Mode 3 is appropriate for acquisition when very precise aiding is available. mode 3 is also used for tracking after acquisition. Mode 3 is also used when the GPS

system is blocked for a very short period of time and needs only to reacquire. Typically, for mode 3 to be appropriate, the time should be known to within 100ms or better.

Position also should also be known to within 30 Km. From a signal processing standpoint, an advantage of mode 3 is that a very small space can be searched. Thus, the search can be very high resolution .

In mode 3, the GPS system is configured so that both a coherent RAM buffer and a non-coherent RAM buffer are dedicated per channel. A relatively short input RAM is used. In this case the length of the input RAM can be much shorter than the length of the PDI. For example, if the PDI is 20ms, only 2 ms of dedicated buffer, which can be filled in ten iterations, is required. In general, however, less code offsets are searched in mode 3 relative to other modes.

In other modes, 4-1 data compression is typically used. In mode 3, data compression is not necessary. 4-bit processing is used. Loss is reduced in comparison to other modes. For example, 1Db of sensitivity is saved by high resolution processing, processing relatively few code offsets, and dedicating coherent and non-coherent buffers. Coherent buffers are saved over a full PDI. Then that PDI is processed to the FFT and added, and the same cycle is repeated through different channels in a circular fashion.

Because the GPS system has a very good idea where to start searching when it decides to use mode 3, the best signal, lowest loss processing can be performed. this includes using the longest coherent integration time the system can afford. Typically, for mode 3 the oscillator is known to .1PPM or .2PPM. In mode 3, updating of position is constantly occurring and uncertainty should be virtually zero. Nonetheless, the GPS system searches to determine whether multipath, auto-correlation, cross-correlation or false peaks are present which might cause false tracking. Once the GPS system is tracking, it typically enters mode 3, is it is not already in mode 3. The GPS system examines peaks, puts several points on the peak and determines whether the result looks like a good signal. As an example, the GPS system may get two signal copies, one direct and one bounced off a building. One of these signal could disappear and the GPS system may track the wrong signal.

An advantage of the GPS system is its ability to put a little window of approximately 10 chips around the output and observe the nearby environment to determine whether there is multipath or cross correlation nearby. In the event the real signal is lost, the GPS system does not become confused, and a possible re-acquisition process is avoided. The GPS system effectively tracks two different paths of one signal with two effective channels built from the different offsets that are constantly being processing. In contrast, a conventional receiver opens a window around this peak, and observes three samples across the peak. The conventional system typically has one correlator as a spare, and it can periodically move that correlator around to look for extraneous signals. The conventional system, however, has no constant visibility. The disclosed GPS system, on the other hand, has the ability to keep the window always open and detect and recover immediately from false tracking.

Mode 3 is typically used as the last step of almost every acquisition. In some case a few strong satellites are found, but not all of the satellites are found. With the strong satellites a position is made, but it may not be the best position. Ideally the available satellites are spread out to minimize the margin of error created by parallax. Therefore, the system begins by finding a few satellites, then minimizes the uncertainty. Mode 3 is entered to make the best measurements possible, make a best position, then possibly find more satellites that were too weak to be found in the other modes.

If ephemeris is available for all of the satellites, the weaker ones can be used for measurements as well. In mode 3, there is also the ability (not shown in the figures) to save processed samples before their magnitude and sum are taken. These are used for conventional tracking.

Choice of Mode

In one embodiment, the processor makes all the decisions at a high level about how the GPS system will operate. The processor's job is to start acquisitions by loading data into the RAM structure. The sequencer's job is basically to manage the flow of data through the GPS system and to cycle through the list, updating state all the while. As the

state updates, the processor looks at it and makes the decision what to do next. The processor makes all the complex calculations for navigation as well as calculating the values for frequency words.

5 The GPS system does try to use what it last knew. The ephemeris is only at best quality for about an hour. It can be used for about four hours, but degrades toward the end. The almanac data is a month-long projection of where the satellites will be. If the GPS system is turned on occasionally, the almanac data may be useable for years, but the satellites move as time goes by. Navigation cannot be performed using only the almanacs.

10 The GPS system, under direction of the processor, bases its initial action son start-up on what it knows, e.g., whether it was turned on an hour ago, whether it has ephemeris on any of the satellites, whether the system was sent a message about it location, etc. All of this information is funneled into a set of initial searches that are programmed into sequences and records.

15 An advantage of the GPS system is that, for an autonomous environment, and given the example of 128K of available RAM, the system can perform tasks tithing a second that older systems required a minute to do. For example, if initially only the almanac data is available, the GPS system can be tracking in a second. If the ephemeris data is available, the GPS system can be navigating in a second with no prior knowledge
20 about its location.

 The processor, based on everything it knows, starts the sequence by typically doing a quick check “everywhere” for a strong signal. The presence of strong signals is noted because they could jam weaker ones. Then any additional known information is used to perform a more sensitive search to find other signals. In some embodiments,
25 there is a limit for complete autonomy, e.g., 27DbHz, to be able to recover the data sequence. The data sequence repeats every thirty seconds, so it takes eighteen seconds to load the block of data needed to perform navigation, but it comes every thirty seconds. The almanac repeats in the remaining twelve seconds. It takes about twelve and one half

minutes to download the entire almanac. Preferably, there is a different source for the almanac, because the system has to be on a rather long time to collect, which has negative consequences for power consumption. Preferably, there is a different source for the ephemeris for similar reasons. However, an advantage of the GPS system is that the
5 signal processing is so powerful that in a conventional environment it can acquire all of the satellites from almost nothing. Then another thirty seconds may be required to make a precise position. Preferably, the almanac and ephemeris data is kept as long as possible before it is collected again. For example, it is kept for one hour and then recollected if necessary. If any of the almanac or ephemeris data is still stored, it is reused to make
10 position in less time, but eventually the data is recollected.

The decisions about how to operate the GPS system are all based on results of previous steps. For example, how many satellites were found, whether the data acquired can be used to find other satellites, etc. The GPS system may perform a different type of search. Typically, a whole sequence may be completed: a cold start first just to see if
15 there are very strong signals; a coarse-aided search based on finding a few of those signals and getting a minimum position; and mode 3 in which only tracking is performed. Even though mode 3 is the most sensitive and lowest loss, and thus yields signals below the data threshold, it is used for tracking because it is the best, most effective tracking mode.

20 As the system is moved around, signals will get strong and weak. If a signal gets strong enough long enough, the ephemeris can be collected, and when it gets weak the ephemeris is used to keep tracking.

Alternative Embodiments

Many alternatives to the specific hardware and software designs shown in the
25 figures and described herein are possible. For example, the RAM, in some systems, is a two-second snapshot, but it's so large at 2 MEG that it is usually off-chip. In an alternative embodiment, a memory interface is included in the GPS system to access an off-chip DRAM through an eight-bit memory. This maintains enough bandwidth for the

signal processor. In another alternative embodiment, non-coherent sums could be stored external to the GPS system.

5 Elements of the control structure, in an alternative embodiment, are replaced with the digital signal processor ("DSP") and part of the processing. For example, a DSP with an FFT core or FFT acceleration could execute a lot of the control logic. With the possible exception of mode 1, the cold start mode in which the data rate is very high, the DSP could handle control functions. The DSP could also take the output of the FFT, rather than the correlator. In other words, the DSP would be in control of the FFT, but the FFT could do the memory fetches on its own, or take the direct path on its own. Also, 10 the sequencing could be done by a DSP. The matched filter could use its internal register for much of its work. It is possible to divide the resources of the matched filter, based on the matched filter design shown. For example, in a full matched filter mode, all 1024 correlators are used for a single chip position. In another matched filter mode, the matched filter resources are divided into halves, quarters or eights, and in each of the 15 resulting matched filter segments a different fractional chip position (e.g., half-chip offsets, quarter-chip offsets, or eighth-chip offsets) is correlated.

In addition, the matched filter resources can be configured to create a high resolution correlation. For example, if the matched filter is divided into two halves, an upper half can be used to correlate two MSBs, while a lower half is used to correlate two 20 LSBs. The two results are then added to generate a higher resolution result. In another example, the matched filter is divided into quarters. The lower two quarters are used to correlate an MSB and an LSB of a high resolution half-chip spacing, and the upper two quarters are used to correlate the MSB and the LSB the other high-resolution half-chip spacing. The division of matched filter resources into parts and use of different parts is 25 completely flexible.

The configuration of the matched filter is determined initially (at setup on a channel-by-channel basis) based on whether a high resolution mode is desired and whether a particular fractional code spacing is desired. The matched filter is configured

U.S. Express Mail No.: ER447520883US
Filing Date: September 2, 2003

PATENT
Docket No. ST02042USV (281-US-P1)

to operate in a particular mode for each channel. The channels access the matched filter in a particular mode on a time multiplexed basis under the control of the sequencer.

APPENDIX A

SERIAL RADIO FREQUENCY TO BASEBAND INTERFACE WITH POWER CONTROL

INVENTORS

STEVEN GRONEMEYER

ROBERT TSO

CROSS REFERENCE TO RELATED APPLICATIONS

[0542] This application is a continuation-in-part of U.S. Patent Application serial number 10/369,853, titled "Serial Radio Frequency to Baseband Interface with Programmable Clock", filed February 19, 2003.

BACKGROUND OF THE INVENTION

[0543] 1. Field of the Invention

[0544] This invention relates to an interface for coupling a radio frequency (RF) processing section to a baseband processing section. More specifically, this invention relates to communicating power control messages from the baseband section to the RF section.

[0545] 2. Related Art

[0546] The worldwide use of wireless devices such as two-way radios, pagers, portable televisions, personal communication system ("PCS"), personal digital assistants ("PDAs") cellular

telephones (also known as “mobile phones”), Bluetooth devices, satellite radio receivers and Satellite Positioning Systems (“SPS”) such as the Global Positioning System (“GPS”), also known as NAVSTAR, is growing at a rapid pace. Current trends are calling for the incorporation of SPS services into a broad range of electronic devices and systems, including PDAs, cellular telephones, portable computers, automobiles, and the like.

[0547] At the same time, manufacturers design their devices using very different architectures, spanning a wide variety of processors, frequency references, clock rates, and the like. The manufacturers are also very interested in keeping costs as low as possible while providing as much functionality (including SPS capability) as possible. In particular, architectures which split SPS signal processing between a radio frequency (RF) front-end a baseband processing section continue to be popular.

[0548] For example, SiRF Technology, Inc. of San Jose, CA made popular an SPS chipset that included the GRF1 RF chip and GSP1/LX baseband processing chip. These two devices are described in detail in the SiRFStar® I GPS Architecture GRF1 and GSP1 data sheets. As shown in Figure 1, the RF chip 102 communicated data samples to the baseband chip 104 using differential sign signal lines (labeled SIGN), differential magnitude signal lines (labeled MAGNITUDE), a GPS clock signal line (labeled GPSCCLK), and an acquisition clock signal line (labeled ACQCLK). The baseband chip 104 could communicate with the RF chip 102 in a limited single purpose fashion, namely, by using automatic gain control (AGC) clock, data, and strobe signal lines (labeled AGCCLK, AGCDATA, and AGCSTRB respectively) to provide AGC data to the RF chip 102.

[0549] More recent SPS signal processing chipset solutions include the SiRFStar® IIe (centered around the GRF2i RF chip and GSP2e baseband chip) and SiRFStar® IIr (centered around the GRF2i RF chip and GSP2t baseband chip) solutions. Both retained the multiple signal lines used to communicate data samples from the RF section to the BB section and the unidirectional communication of AGC information from the baseband section to the RF section. However, the BB section communicated AGC information unidirectionally to the RF chip using a single pulse width modulated output that the RF chip sampled. In other words, the SiRFStar® IIe eliminated the multiple signal line AGC communication path in favor of a single output line.

[0550] For power control, the RF chip typically included a dedicated power control input, for example, one power control input pin that would enable or disable the majority of the RF chip. Thus, there was little or no ability to exercise detailed control over the power consumed by the RF chip. In other words, when the RF chip was active, so were most of the hardware blocks (e.g., phase locked loops, frequency dividers, digital interface sections, and the like) in the RF chip, whether they were needed at the time or not. As a result, the RF chip would consume greater average power than was otherwise necessary. Particularly when incorporated into a device with limited power reserves, such as a battery operated GPS receiver, excess power consumption was a significant drawback.

[0551] Therefore, a need exists to overcome the problems noted above and others previously experienced.

SUMMARY

[0552] The invention provides RF power control messaging, as well as related methods of providing RF power control messaging, over an interface between an RF processing section and a baseband processing section. The interface supports general purpose bi-directional message transmission between the RF processing section and the baseband processing section. The interface further supports transmission of SPS signal samples between the two processing sections without adding undue complexity to the interface.

[0553] In one implementation, the interface includes a message serial interface and a data serial interface. The data serial interface communicates SPS signal sample data from the RF section to the baseband section. The message serial interface communicates messages, including power control messages, between the RF section and the baseband section.

[0554] As noted above, a message serial interface communicates power control messages between the processing sections. The message serial interface may include a message-in signal line, a message-out signal line, and a message clock signal line. In some implementations, the message serial interface may also include a slave-select signal line. A power control message may include, for example, multiple power control bits. Each power control bit may specify a power state (e.g., powered-up or powered-down) for pre-determined circuitry in the RF section.

[0555] The complexity of the data serial interface may be reduced, for example, by using a single data bit signal line to serially carry signal samples from the RF section to the baseband section. The data serial interface may also include a data clock signal line that provides timing for

the signal samples. In particular, as an example, the data clock signal line may carry a data clock (that includes rising edges and falling edges) nominally running at $16 f_0$, where $f_0 = 1.023$ MHz, while the data bit signal line may carry a data signal comprising serially transmitted data bits. In one implementation, a first type of data bit is valid on the rising edge of the data clock and a second type of data bit is valid on the falling edge of the data clock. As an example, the first type of data bit may be a sign bit, while the second type of data bit may be a magnitude bit.

[0556] Other apparatus, systems, methods, features and advantages of the present invention will be or will become apparent to one with skill in the art upon examination of the following figures and detailed description. It is intended that all such additional systems, methods, features and advantages be included within this description, be within the scope of the present invention, and be protected by the accompanying claims.

BRIEF DESCRIPTION OF THE FIGURES

[0557] The components in the figures are not necessarily to scale, emphasis instead being placed upon illustrating the principles of the invention. In the figures, like reference numerals designate like parts throughout the different views.

[0558] Figure 1 of Appendix A shows a prior interface between a GPS RF chip and a baseband chip.

[0559] Figure 2 of Appendix A illustrates a satellite positioning system receiver that includes an RF processing section coupled to a baseband processing section by an interface that includes a message serial interface and a data serial interface.

[0560] Figure 3 of Appendix A illustrates a timing diagram that shows the relationship between a data clock and a data signal carried, respectively, on a data clock signal line and a data bit signal line that form the data serial interface shown in Figure 2.

[0561] Figure 4 of Appendix A illustrates a timing diagram that shows the relationship between a message clock and message data bits carried, respectively, on a message clock signal line and message data bit signal line that form part of the message serial interface shown in Figure 2.

[0562] Figure 5 of Appendix A shows a method for interfacing an RF processing section and a baseband processing section.

DETAILED DESCRIPTION

[0563] A typical satellite positioning system ("SPS") system has approximately 12 satellites that may be visible at any one time to a wireless device. As used in this document, SPS means any system utilizing satellites and/or land-based communications devices for providing or enabling the determination of a location of the wireless device on the earth, including, but not limited to: a global positioning system ("GPS") (such as NAVSTAR), GLONASS, LORAN, Shoran, Decca, or TACAN. For the purposes of discussion, specific examples of an interface between a GPS RF processing section and a baseband processing section are described. However, the principles underlying the interface are applicable to interfacing RF processing and baseband processing sections in general.

[0564] Turning first to Figure 2 of Appendix A, that figure shows a receiver 200 of a satellite positioning system. The receiver 200 includes an RF processing section 202 coupled to a baseband

processing section 204 using an RF-to-baseband interface 206. The RF processing section 202 receives SPS signals, for example the 1575.42 MHz GPS signal, on the RF input 207.

[0565] The receiver section 200 may be generally regarded as including an RF front end 224 and a baseband back end 226. The RF front end 224 includes the RF processing section 202 and RF-to-baseband interface 206. The RF front end 224 processes the SPS signals received on the RF input 207 through a sequence of downconversion, automatic gain control, and analog to digital conversion. The baseband back end 226 includes the baseband processing section 204 and RF-to-baseband interface 206. The baseband back end 226 processes (using a microcontroller core, CPU, or other control logic) the sampled data provided by the RF front end 224. The baseband back end 226 communicates the processed data to a digital device (e.g., a digital signal processor, general purpose microcontroller or CPU, or host PC) using one or more address, data, control, and clock signals that comprise the digital communication interface 222.

[0566] Either or both of the RF front end 224 and baseband back end 226 may be implemented as individual single integrated circuits, for example. Thus, the RF front end 224 may be a single package that includes the RF input 207 (e.g., a particular input pin on the package), RF processing section 202, and interface 206 (e.g., a set of interface pins as described in more detail below). Similarly, the baseband back end 226 may be a single package that includes the baseband processing section 204, interface 206, and digital interface 222. The processing performed by the RF processing section 204 and baseband processing section 204 may be that set forth in more detail in the SiRFStar® I, II, or III chipset data sheets, while the interface 206 is described in more detail below. The SiRFStar® chipsets are available from SiRF Technology, Inc. of San Jose California.

[0567] Although, as shown in Figure 2 of Appendix A, the functional division between the RF front end 224 and the baseband backend 226 lends itself to being divided into two separate integrated circuits, many other implementations are possible. As one example, numerous discrete logic and signal processing circuit blocks may implement the RF, baseband, and interface 206 functionality. As additional examples, any of the circuitry underlying the functionality of the RF front end 224 and the baseband back end 226 may be incorporated into a single package (e.g., that encloses multiple integrated circuit dies) or integrated circuit, multiple packages or integrated circuits, or distributed across one or more circuit boards. In these implementations, individual wires, circuit board traces, or VLSI metal or polysilicon layers carry the interface 206 signals between the RF processing circuitry and the baseband processing circuitry.

[0568] Furthermore, any of the circuitry underlying the functionality of the RF front end 224 and the baseband back end 226 may be incorporated, with additional functionality, into a single package or integrated circuit, multiple packages or integrated circuits, or distributed across one or more circuit boards. As examples, the RF and baseband circuitry may be integrated on a die with digital or analog processing circuitry for cellular telephony, PDA operation, or engine, instrument, or electronics controllers for automobiles. Thus, Figure 2, and the examples given above are not limiting; rather, one of ordinary skill in the art will appreciate that the particular implementation, division of functionality, and packaging of the circuitry that implements the RF processing, baseband processing, and interface 206 may vary widely depending on the application at hand, engineering considerations, cost considerations, and the like.

[0569] The interface 206 includes a message serial interface 208 and a data serial interface 210. The message serial interface 208 provides for serial communication of general purpose messages bi-directionally between the RF section 202 and the baseband section 204. In contrast, the RF section 202 employs the data serial interface 210 to transmit SPS signal samples to the baseband section 204.

[0570] As an initial matter, it is noted that in general, the interface 206 signals shown in Figure 2 are CMOS compatible. In particular, the inputs, for logic one, are above $0.7 \cdot V_{cc}$ V, and, for logic zero, are below $0.3 \cdot V_{cc}$ V. Outputs, for logic one, are above $V_{cc} - 0.4$ V, and, for logic zero, are below 0.4 V. The input/output pins generally operate in either the 2.5 V or 3.3 V voltage ranges, depending on the desired implementation. The real time clock (RTC) input/output pins may operate at 1.5 V, although they may be designed to tolerate 3.3 V levels if desired. Any of the signals, however, may be adapted to different voltage ratings or specifications depending on the desired implementation.

[0571] The message serial interface 208, as shown in Figure 2, includes the message-in signal line (labeled MSG_DO / MI), a message-out signal line (labeled MSG_DI / MO), a message clock signal line (MSG_CLK / MK) and a slave-select signal line (labeled MSG_CEB / SS_N[0]). The labels on the message signal lines indicate the direction of data flow from the perspective of the RF section 202 / baseband section 204. For example, the message-out signal line (MSG_DI / MO) carries message bits input to the RF section 202 and output by the baseband section 204.

[0572] The data serial interface 210 includes the data clock signal line (labeled ACQCLK) and the data bit signal line (labeled SGNMAG). The data serial interface 210 generally uses only a

single data bit signal line to communicate, serially, data bits to the baseband section 204 (as discussed below in greater detail with regard to Figure 3). Thus, the data serial interface 210 generally includes as few as two signal lines: one for a data clock and one for data bits. The data serial interface 210 is thus a low complexity solution for a SPS signal sample interface between the RF section 202 and the baseband section 204.

[0573] As shown in Figure 2 of Appendix A, the receiver section 200, on the RF processing side, also includes a real time clock (RTC) oscillator (OSC) and monitor section 212. A 32 KHz crystal (or other clock source) provides an input clock 214 for the RTC OSC section 212. The RTC OSC section 212 generates a clock output on the RTCLK / RIN signal line that the baseband section 204 uses to keep, as examples, GPS time or UTC time. The clock output is, for example, a 32,768 Hz 1.5 V CMOS output. The RTC OSC section 212 continues to run during power down modes to help the baseband section 204 maintain an accurate timebase.

[0574] However, monitoring circuitry (e.g., a rectifier coupled to the clock input and followed by a comparator) in the RTC OSC section 212 determines when the input clock 214 has consistently run (e.g., has stopped for no more than 10-30 clock cycles). If the clock has stopped for too long, then the RF section 202 sets a bit (e.g., sets a flip/flop output or sets a bit in a multi-bit status register) to indicate that the clock output has not been consistent (and, in some cases, that the baseband section 204 should search over the full range of the received SPS signal to determine the correct time).

[0575] The RF section 202 also accepts clocking input from either a crystal oscillator 216 or an external clock source 218 (e.g., a frequency reference provided in a wireless device). The clocking

inputs 216 and 218 provide a clock source that a PLL divider chain in the RF section 202 uses to generate the ACQCLK signal. The clocking inputs 216 and 218 are collectively referred to below as the OSCCLK, while the PLL divider chain clock is referred to as the PLLCLK. The PLLCLK is typically set to generate a nominal frequency of $16 f_0$ (where $f_0 = 1.023$ MHz) on the data clock ACQCLK derived from the OSCCLK (or an internal reference).

[0576] At power-up, the OSCCLK (generally in the range of 5 - 27 MHz) is present on the ACQCLK output. A message (described below) commands the RF section 202 to switch ACQCLK from OSCCLK to the PLLCLK and from the PLLCLK to the OSCCLK. The ACQCLK signal may be a 2.5 / 3.3 V CMOS output with a duty cycle between 45% and 55% (except when switching clock sources, in which case ACQCLK may have an extended low cycle).

[0577] A power control signal (labeled PWRUP / RFPWRUP) may optionally be provided to control whether certain portions of the RF section 202 are powered-up. The power control signal may be connected, for example, to a voltage regulator enable pin in the RF section 202 to provide a coarse power-up / power-down control over the majority of the circuitry in the RF section 202. On the other hand, the RTC OSC section 212 is separately powered so that it can continue to provide a clock to the baseband section 204. The power control signal may be a 2.5 / 3.3 V CMOS signal. The baseband processing side includes an RTC logic section 220. The RTC logic section 220 accepts the input clock generated by the RTC OSC and monitor section 212 as an aide in determining the current time as well as SPS location solutions.

[0578] The RTC logic section 220 also outputs the reset signal GRFRST_N / RESET_N (asserted low). The reset signal may be used to reset the state of control registers in the RTC OSC

section 212 and the RF section 202 at power-on. For example, when GRFRST_N is asserted, the digital control registers on the RF processing side will be reset to their default states. The default states of the control registers allow the OSCCLK circuits to operate and allow the ACQCLK output to be driven by OCSCLK (when PWRUP) is asserted. When GRFRST_N is not asserted, then the RF section 202 operates according to its internal logic states.

[0579] In one implementation, the message serial interface signals are 2.5 / 3.3V CMOS I/O signals. The MSG_CLK / MK, MSG_DI / MO, and MSG_CEB / SS_N[0] signals are inputs to the RF section 202. The MSG_DO / MI signal is an output from the RF section 202 with tri-state control. When the MSG_CEB / SS_N[0] is logic high, the MSG_DO / MI output is high impedance and may be driven by other devices that are also connected to the message serial interface 208. Thus, the MSG_CEB / SS_N[0] output from the baseband section 204 operates as a slave selection signal that allows the RF section 202 to drive data on the MSG_DO / MI signal line. When additional devices are attached to the message serial interface 208, the baseband section 204 may provide additional slave selection signal lines to determine which device is allowed to drive data on the MSG_DO / MI signal line.

[0580] The RF section 202 may also include one or more inputs for external analog sensors (not shown). Thus, a multi-channel analog to digital (A/D) converter in the RF section 202 may take measurements of analog input signals and communicate the results to the baseband section 204. The analog inputs may include, but not be limited to, temperature inputs, gyro turn rate inputs, wheel tick inputs, or a battery voltage inputs.

[0581] Table 1 summarizes the operating modes for the receiver section 200:

Table 1			
Mode	GRFRST_N	PWRUP	Operation
Sleep	0	0	RF section voltage regulator disabled; RTC OSC section isolated from RF section.
Start-up	0	1	RF section voltage regulator enabled; RTC OSC section isolated; registers reset; OSCCLK enabled; ACQCLK outputs OSCCLK.
NA	1	0	Not allowed.
Normal	1	1	RF section voltage regulator enabled; RTC OSC section communicates with RF section; messages control RF section operation.

[0582] Turning next to Figure 3 of Appendix A, that figure illustrates a timing diagram 300 that shows the relationship between a data clock 302 and a data signal 304. The data signal 304 provides SPS signal samples to the baseband section 204. The SPS signal samples are derived from an SPS input signal received by an antenna connected to the RF section 202. The ACQCLK signal line carries the data clock 302, while the SGNMAG signal line carries the data signal 304. The data signal 304, which may be, for example, a 2.5 / 3.3 V CMOS output, transmits both sign bit data 306 and magnitude bit data 308 on the SGNMAG signal line. In one implementation, the data signal 304 provides sign and magnitude bit information determined by an A/D converter in the RF section 202.

[0583] In other implementations, additional bits of information or quantization may be provided, in concert with a predetermined protocol or encoding technique applied to the data bits (e.g., a pseudorandom noise code) to allow the baseband section 204 to identify the data

transmitted. Furthermore, the data signal 304 may transmit signal samples for different radio chains handled by the RF section 202. For example, when the RF section 202 is processing SPS data, the data signal 304 may bear the two bits per sample (sign and magnitude) data pairs noted above. In contrast, when the RF section 202 is processing a different RF signal (e.g., a Bluetooth signal), the data signal 304 may instead transmit more or less bits per sample (e.g., 4 or 6 bits) in accordance with the guidelines established for processing that RF signal. Similarly, the data clock 302 may vary in frequency and duty cycle to meet the processing guideline for the RF signal that the RF section 202 is currently processing.

[0584] As shown in Figure 3 of Appendix A, the RF section 202 outputs the sign bit 306 when the data clock 302 is high and outputs the magnitude bit 308 when the data clock 302 is low. As shown in Figure 3, the sign bit 306 is valid no less than $T_{\text{SETUP-F}}$ before the falling edge 310 of the data clock 302. Similarly, the magnitude bit 308 is valid no less than $T_{\text{SETUP-R}}$ before the rising edge 312 of the data clock 302.

[0585] The sign bit 306 remains valid no less than $T_{\text{HOLD-F}}$ after the falling edge 310 of the data clock 302. The magnitude bit 308 remains valid no less than $T_{\text{HOLD-R}}$ after the rising edge 312 of the data clock 302. The setup and hold times may vary from implementation to implementation. As one example, the setup and hold times may be approximately 5 - 10 ns.

[0586] The message serial interface 208 may be implemented in a wide variety of ways. In one implementation, the message serial interface 208 has the characteristics set forth below, although other implementations are also possible.

[0587] The message serial interface on the RF section 202 operates as a slave device to the baseband section 204 (or other master device that adheres to the characteristics set forth below). The inputs bits to the RF section 202 (on the MSG_DI line) are shifted into a 32 bit shift register in the RF section 202 under control of the MSG_CLK. In one implementation, up to 32 bits are sent in one message block and data is received and transmitted with the most significant bit first. Simultaneously, the MSG_DO output bits are shifted out of the other end of the same shift register. If output from the RF section 202 is not needed, then the MSG_DO output need not be connected. In one implementation, the MSG_CLK operates at up to 20 MHz and the message serial interface signals are, approximately, above $0.8 \cdot V_{CC}$ V for logic 1 and below $0.2 \cdot V_{CC}$ V for logic 0.

[0588] The slave select signal line (MSG_CEB) is active low for serial data transmission. The MSG_DI and MSG_CLK may therefore be ignored as long as MSG_CEB has been high for a pre-selected period of time (e.g., 5 ns). Data is sampled on the rising edge of MSG_CLK. In one implementation, a transition on MSG_DI or MSG_DO occurs at least 5 ns after the rising edge of MSG_CLK and stabilizes at least 5 ns before the next rising edge of MSG_CLK. The data is shifted on the falling edge of MSG_CLK. Continuing the example, the MSG_CEB signal may be active (logic 0) at least 10 ns before the rising edge of the first MSG_CLK and may remain active (logic 0) at least 10 ns after the last falling edge of MSG_CLK. The time interval in both cases may be, for example, one half of one clock cycle. The MSG_CEB signal may then be held inactive (logic 1) for at least 30 ns to provide time for the RF section 202 to latch the data.

[0589] If the MSG_CEB signal transitions high before all data in a message block have been sent, the data is discarded and not applied to RF section 202 registers. Unused bits in a message

block are set to zero. However, a fast write mode is provided to allow for a shortened, one byte message. The fast write mode is assumed until more than 8 bits have been received. When more than 8 bits have been received, the RF section 202 expects to receive a full 32 bits for a valid message.

[0590] The RF section 202 outputs data (on MSG_DO) to the baseband section 204 in response to a message received from the baseband section 204 that requests the data. The baseband section 204 then sends a subsequent message to shift out the requested data in the RF section 202 shift register from the shift register. The subsequent message may be an independent operational message or it may be a dummy message sent for the sole purpose of shifting out the desired data.

[0591] Figure 4 of Appendix A illustrates a timing diagram 400 that shows the relationship between the slave select signal (MSG_CEB) 402, the message clock signal (MSG_CLK) 404, and the message data bit signals (MSG_DO and MSG_DI) 306. Data transmission starts when the slave select signal 402 falls. The transmitted data are latched when the slave select signal 402 rises.

[0592] As shown in Figure 4 of Appendix A, the message-out signal line (MSG_DI / MO) and message-in signal line (MSG_DO / MI) each carry a serial bit stream. The serial bit stream on the message-out signal line represents a message selected from a group of predefined RF section messages that are transmitted from the baseband section 204 to the RF section 202. Similarly, the serial bit stream on the message-in signal line represents a message selected from a group of predefined baseband section messages that are transmitted from the RF section 202 to the baseband section 204.

[0593] The messages are not limited to any particular purpose or format. As explained in more detail below, the messages may include, but not be limited to, RF section power control messages, RF section test messages, clock status messages, analog measurement messages, channel conversion count messages, and the like.

[0594] In one implementation, there are four types of message blocks defined. Data [1:0] (in a 32-bit or 8-bit sequence) are address bits that define the four messages as shown in Table 2 below. Each message type is able to support both a fast write mode and a full write mode and spare capacity has been defined for both modes.

Table 2 - Message Blocks		
Data [1:0]	Message Type	Message Name
00	0	AGC (Fast Write) and Synthesizer
01	1	Power Control (Fast Write) and Synthesizer
10	2	Output Message Types
11	3	Input Message Type Expansion

[0595] Exemplary contents of each message are shown in detail in Tables 3 through 8. Table 3 shows AGC and synthesizer control messages, Table 4 shows power control and synthesizer control messages, and Table 5 shows output request types for a selected message type. Table 6 shows output message types and Tables 7-8 show input message types. The columns define the contents as follows. The first column, labeled Bits, represents the message data bits, with bit 0 indicating the last bit transmitted. The second column, labeled Field Name, identifies the name of the field in the message. The third column, labeled Length, is the length of the field. The forth column, labeled Default, indicates the contents of the default parameter in the RF section 202 when power is first

applied. The fifth column, labeled Contents, describes the allowed contents of the field. The sixth column, labeled Function, indicates what the field accomplishes. The seventh column, labeled _Pwr, indicates which power domain control bit shown in Table 4, if any, is used to drive these field outputs to zero on the interface to the RF section 202.

[0596] Message type 2 provides for implementing output requests using a field that specifies up to 32 types of output requests. Message type 3 provides for expanding the input message types (or addresses) from 4 to 36. References below to the "synthesizer" are references to the PLL synthesizer clock generation circuitry in the RF section 202. The PLL synthesizer is configurable, for example, by setting clock divider values to generate the PLLCLK from a number of different input reference frequencies.

Table 3 – Message Type 0: AGC and Synthesizer Control (Address [1:0] = 0)						
Bits	Field Name	Length	Default	Contents	Function	_Pwr
31:28	Spare0 [3:0]	4	0	0	Spare	
27:8	NUM [19:0]	20	TBD	0x00000 – 0xFFFFF	Specifies the numerator of the fractional part of the loop divider in the PLL clock generation section of the RF section 202.	Synth
7:2	AGC [5:0]	6	0	0x00- 0x3F	Controls AGC gain in the RF section 202	Rx
1:0	Address [1:0]	2	0	0	Defines message type	

Table 4 – Message Type 1: Power Control and Synthesizer Control (Address [1:0] = 1)						
Bits	Field Name	Length	Default	Contents	Function	_Pwr
31:28	Spare1 [3:0]	4	0	0	Spare	
27	InvertFePwr	1	0	0: Fe_Pwr = Rx_Pwr 1: Fe_Pwr = ~Rx_Pwr	Partition the reception chain in the RF section 202 for testing purposes	
26	WideBwFilter	1	1	0 = Narrow BW 1 = Wide BW	Select the filter used in the RF section 202.	Rx
25:18	ND [7:0]	8	-	0x00 – 0xFF	Specifies the integer part of a synthesizer loop divider parameter in the PLL clock generation section	Synth
17:15	RDIV [2:0]	3	-	0x0 – 0x7	Specifies a synthesizer reference divider value in the PLL clock generation section	Synth
14:11	CP [3:0]	4	-	-	Specifies the synthesizer charge pump output and test modes	PLL
10	PD_POL	1	-	1=positive, 0=negative	Specifies the phase detector polarity	PLL
9	DvSel	1	1	0=Fractional 1=Integer	Specifies the divider for PLL	PLL

Table 4 – Message Type 1: Power Control and Synthesizer Control (Address [1:0] = 1)						
Bits	Field Name	Length	Default	Contents	Function	_Pwr
					feedback	
8	SDO	1	1	0=Third Order SD 1=First Order SD	Chooses Sigma Delta Order	Synth
7	Rx_Pwr	1	0	1=on, 0=off	Controls front end power for 2 nd low noise amplifier through A/D converter	
6	AcqClk_Sel	1	0	1= PLL, 0=Osc	Controls glitch-free switch that selects OSCCLK or PLLCLK for ACQCLK	
5	Synth_Pwr	1	0	1=on, 0=off	Controls power to fractional N synthesizer	
4	PLL_Pwr	1	0	1=on, 0=off	Controls power for PLL and divider chain	
3	LNA1_Pwr	1	0	1=on, 0=off	Controls power for first (optional) LNA	
2	Osc_Pwr	1	1	1=on, 0=off	Controls power for oscillator, ACQCLK-select mux and ACQCLK driver	
1:0	Address [1:0]	2	1	1	Defines message type	

Table 5 – Message Type 2: Output Request Types 0 to 31 (Address [1:0] = 2)					
Bits	Field Name	Length	Default	Contents	Function
31:8	Spare2 [28:5]	24	0	0	Spare
7:3	Spare2 [4:0] or Out_Dat [4:0]	5	0	0-31	Spare (Fast Write), if Out_Req=0 Output data type, if Out_Req=1
2	Out_Req	1	0	0=data 1=output	When=0, data follows When=1, output data to load follows
1:0	Address [1:0]	2	2	2	Defines message type.

[0597] Output message types are shown in Table 6. Spare messages have been defined for expansion or use in testing the RF section 202. Since this data is input to the message interface from the RF section 202, these fields are given names denoting input, such as spareInA. When the data is shifted out, it is positioned in the output data stream using the index values given. For example, spareInA [23:0] would be located in the final 24 bits shifted out in the 32 bit output field, so that eight leading zeros would be followed by spareInA [23] through spareInA [0] according to the convention of shifting out the most significant bit first.

[0598] Out_Dat [4:0] = 4-8 specify 20-bit measurements taken by a dual slope A/D converter in the RF section 202. As noted above, the A/D converter may have multiple channels connected to one or more analog measurement devices. As used below, Out_Dat [4:0] = 9 specifies the valid clock bit maintained by the RTC OSC section 212 and described above.

Table 6 – Output Messages Defined Using Message Type 2 (Address [1:0] = 2 and Out_Req = 1)		
Out_Dat [4:0]	Message Bits	Contents
0	31:24 23:0	0 SpareInA [23:0]
1	31:24 23:0	0 SpareInB [23:0]
2	31:24 23:0	0 SpareInC [23:0]
3	31:24 23:0	0 SpareInD [23:0]
4	31:30 29:20 19:0	DS_ADC_CH_0 LAST_CH [1:0] Spare [9:0] DATA0 [19:0]
5	31:30 29:20 19:0	DS_ADC_CH_1 LAST_CH [1:0] Spare [9:0] DATA1 [19:0]
6	31:30 29:20 19:0	DS_ADC_CH_2 LAST_CH [1:0] Spare [9:0] DATA2 [19:0]
7	31:30 29:20 19:0	DS_ADC_CH_3 LAST_CH [1:0] Spare [9:0] DATA3 [19:0]
8	31:30 29:20 19:0	DS_ADC_CNT LAST_CH [1:0] Spare [9:0] COUNT [19:0]
9		RTC_STA

Table 6 – Output Messages Defined Using Message Type 2 (Address [1:0] = 2 and Out_Req = 1)		
Out_Dat [4:0]	Message Bits	Contents
	31:1 0	Spare [30:0] 0=RTC Not Valid (default) 1=RTC Valid
29	31:26 25:0	0 Power Control Message Register [24:0]
30	31:0	Message Input Shift Register
31	31:16 15:0	0 Chip Version [15:0]

Table 7 – Message Type 3: Expanded Input Message Types (Address [1:0] = 3)					
Bits	Field Name	Length	Default	Contents	Function
31:8	Spare3 [28:5]	24	0	0	Spare
7:3	Spare3 [4:0] or Address [6:2]	5	0	0-31	Spare (Fast Write), if Addr_Exp=0 Address expansion, if Addr_Exp=1
2	Addr_Exp	1	0	0=data 1=address	When=0, data follows When=1, address follows
1:0	Address [1:0]	2	3	3	Defines message type.

[0599] Spare messages (useful for test or expansion purposes) are shown in 8. Because these data represent control bits output by the message interface to the RF section 202, the data fields have been named to denote outputs, for example SpareOutA.

[0600] A test message has also been define for the SGNMAG output signal line. When TestSignMag [8] =.one, the test mode is entered. When the bit is a zero, test mode is turned off. In

test mode, the pattern specified in TestSignMag [7:0] is output, as long as ACQCLK is running, beginning with TestSignMag [7] while ACQCLK is high.

Table 8 – Input Messages using Message Type 3 (Address [1:0] = 3 and Addr_Exp = 1)			
Address [6:2]	Msg Bits	GRFRST_N	Contents
0	31:8	0	SpareOutA [23:0]
1	31:8	0	SpareOutB [23:0]
2	31:8	0	SpareOutC [23:0]
3	31:8	0	SpareOutD [23:0]
4	16:8	0	TestSignMag [8:0]
5			DS_ADC_PER
	31:12	0	PERIOD [19:0]
	11	0	CLK_SEL
	10	0	CLK_ENB
	9:8	0	Spare [1:0]
6			DS_ADC_SH
	31:12	0	SHIFT [19:0]
	11:8	0	Spare [3:0]
7			DS_ADC_PH
	31:12	0	PH_ONE [19:0]
	11:8	0	Spare [3:0]
8			DS_ADC_SEQ
	31:12	0	CH_SEQ [23:0]
	11:8	0	Spare [3:0]
9			RTC_CTL
	31:10	0	Spare [21:0]
	9	0	1: Set RTC Status 0: No action (default)
	8	0	1: Read RTC Status 0: No action (default)
10			SGNMAG_SIG

Table 8 – Input Messages using Message Type 3 (Address [1:0] = 3 and Addr_Exp = 1)			
Address [6:2]	Msg Bits	GRFRST_N	Contents
	31:1 0	0 0	Spare [30:0] 0: SGNMAG (default) 1: OSCCLK

[0601] Address [6:2] = 5 - 8 specify parameters for the dual slop A/D converter in the RF section 202. The DS_ADC_PER message sets the 20-bit conversion period, the duration of an entire A/D conversion cycle (PERIOD), selects one of the input clocks (e.g., OSCCLK or PLLCLK) provided to the A/D converter (CLK_SEL), and enables or disables the clock (CLK_ENB). The DS_ADC_SH message provides a 20-bit shift period (SHIFT) that the A/D converter control circuitry uses as a count down value before initiating conversion in order to change the phase of the A/D conversion cycles relative to any given time base. The DS_ADC_PH specifies a 20-bit phase one conversion period (e.g., the duration of the integration period) for the dual slop A/D converter.

[0602] The DS_ADC_SEQ message specifies 24 bits that control the order in which the A/D converter performs a conversion on each of four input channels. More specifically, the 24 bits are partitioned into 12 pairs of bits; each pair specifies the next input channel to the analog multiplexer before the A/D converter. The pairs of bits thus control which channel is next digitized by the A/D converter and the four input channels may thereby be sampled at different rates.

[0603] Continuing with regard to Table 8, Address [6:2] = 9 specifies that the clock status bit in the RTC OSC section 212 will be set to indicate a good clock, or (if bit 8 is a one) that the baseband section 204 is requesting the value of the clock status bit to be output by the RF section 202. Address [6:2] = 10 controls (e.g., via a multiplexer) the signal that the RF section 202 provides on the SGNMAG signal line. The default is the sign bit and magnitude bit information, while the alternative is the OSCCLK signal.

[0604] Bits 2-7 in the message format shown in Table 4 are power control bits. Those bits control whether particular hardware elements in the RF section 202 are powered-up or powered-down. The bits may be transferred from the shift register in the RF section 202 and applied to power control circuits that apply or remove power from specific hardware elements. For example, bit 5, the PLL_Pwr bit, controls power for a phase locked loop (PLL) circuit and frequency divider in the RF section 202. When the RF section 202 receives a message that has bit 5 cleared, the RF section 202 may remove power from the PLL and divider circuitry by opening a switch through which power flows, driving a power regulator control pin, or through another mechanism. Similarly, when the RF section 202 receives a message that has bit 5 set, the RF section 202 may apply power to the PLL and divider circuitry by closing the switch, enabling the power regulator, or the like.

[0605] While the message format provides five power control bits for (2, 3, 4, 5, and 7), more or fewer power control bits may be provided depending on the implementation. Each bit specifies a power control state (e.g., power-up or power-down) for one or more sections of pre-selected circuitry in the RF section 202. Furthermore, in other implementations, multiple bits may be

employed to specify a power state that includes multiple levels of power control. Thus, for example, two bits may be employed to specify one of four different power states for a particular set of circuitry in the RF section 202.

[0606] While Tables 2-8 provide one example of message formats, many other implementations are also possible. Shown below in Tables 9-11 is another exemplary format that employs 56-bit messages without using a 2-bit message block definition.

Table 9 - Message Structure					
Bit # <0:55>	Field Name	Length (bits)	Contents	Function	Default
55	Tst_Ref_Div	1	0 = normal operation 1 = connect reference divider output to test output pin (e.g., scan data output).	Reference divider scan test	0
54	ID_Read	1	0 = normal operation 1 = revision number output to test output pin (e.g., scan data output).	Chip ID read function	0
53	IF_TestMux	1	0 = AGC Test Point 1 = Mixer Test Point	Selects either AGC Test Point or Mixer Test Point to TP_IF pin if Mode is set to 11 (IF Test Point Enable).	0
52-29	NUM[23:0]	24	x000000 to xFFFFFF	Numerator of the fractional part of the loop divisor	x898232
28-21	ND[7:0]	8	x00 to xFF	Synthesizer integer part of loop divider	x5A
20-18	SPARE	3	(default)	Not Used	x0

Table 9 - Message Structure

Bit # <0:55>	Field Name	Length (bits)	Contents	Function	Default
17-14	CP[3:0]	4	See Table 11	Synthesizer Charge pump output current, and test modes	1011
13	PD_POL	1	1 = positive 0 = negative	Phase detector polarity	1
12-11	ACC[1:0]	2	00 = 3 Fractional Accumulators 01 = 2 Fractional Accumulators 10 = Integer Divider, 0 Accumulators 11 = Integer Divider, 0 Accumulators		00
10	SPARE	1	Pad with zeros	Not Used	1
9	CMOS_PEC LB	1	1 = CMOS output buffer selected 0 = PECL output buffer selected	Selects the CMOS or PECL output buffer.	0
8	DIV32_EN	1	1 = enable 0 = disable	Divide by 32 enable	1

Table 10 - FAST Mode Bits - Power Control

Bit # <0:55>	Field Name	Length (bits)	Contents	Function	Default
7	Ref_Osc_EN	1	1 = enable 0 = disable	Power control for the clock oscillator and buffer section in the RF section 202	1
6	RX_Chain_EN	1	1 = enable 0 = disable	Power control for the radio frequency amplifier (RFA), Mixer, AGC amp, and ADC in the RF section 202. Enables ACQCLK, SIGN, and	1

				MAG outputs.	
5	LNA_EN	1	1 = enable 0 = disable	Power control for the LNA Amplifier circuit in the RF section 202.	1
4	CLKGPS_EN	1	1 = enable 0 = disable	Power control for the CLKGPS, and PECL reference if PECL is selected.	1
3	Synth_EN	1	1 = enable 0 = disable	Power control for the Charge Pump, Phase Detector, Prescaler, and Logic in the RF section 202.	1
2	VCO_EN	1	1 = enable 0 = disable	Power control for the VCO circuit in the RF section 202	1
1-0	Mode[1:0]	2	00 = GPS Clock Only Mode 01 = Normal Operating Mode 10 = Standby Mode (Sleep) 11 = IF Test Point Enable	Sets up the operating mode of the RF section 202.	00

Table 11 - Charge Pump Programming Fields

cp<3>	cp<2>	cp<1>	cp<0>	
1	0	0	0	50 uA, Normal operation of charge pump
1	0	0	1	100 uA, Normal operation of charge pump
1	0	1	0	300 uA, Normal operation of charge pump
1	0	1	1	500 uA, Normal operation of charge pump
1	1	0	0	700 uA, Normal operation of charge pump
1	1	0	1	900 uA, Normal operation of charge pump
0	0	1	1	Test mode: all charge pump outputs source current
0	0	0	1	Test mode: all charge pump outputs sink current
0	1	1	1	Test mode: all charge pump outputs source and sink current simultaneously.

[0607] Tables 9-11 show an implementation in which the last eight bits of the message are power control bits. Thus, power control may be communicated through a fast message as set forth previously. As noted above, the power control bits determine whether particular hardware elements in the RF section 202 are powered-up or powered-down. The power control bits are not limited to controlling the hardware blocks described in Table 10 (or Table 4). Instead, depending on the implementation, the power control bits may be established in the message to control power to any desired hardware circuitry that will be incorporated into the RF section 202.

[0608] The baseband section 204 may thereby establish a detailed control over the power consumed by the RF section 202. In other words, the baseband section 204 may determine, at any given time, those hardware blocks in the RF section 202 that will operate, and those that will be powered down. As a result, the RF section 202 will consume less average power than an RF section in which all the hardware blocks operate continuously. Such power control is very useful in battery operated devices, or in any other SPS enabled device with a limited power supply.

[0609] Turning next to Figure 5 of Appendix A, that Figure shows a flow diagram 500 that shows a method for interfacing the RF section 202 and the baseband section 204. In particular, with regard to the serial transmission of SPS signal samples to the baseband section 204, the RF section 202 places a sign bit 306 on the SGNMAG signal line (step 502), then provides a falling edge 310 on the ACQCLK line (step 504). Subsequently, the RF section 202 places a magnitude bit 308 on the SGNMAG signal line (step 506), then provides a rising edge 312 on the ACQCLK line (step

508). This sequence repeats for each sign bit and magnitude bit sample pair transmitted to the baseband section 204. SPS signal data is thereby serially transferred to the baseband section 204.

[0610] With regard to message transmission between the RF section 202 and the baseband section 204, the master device (typically the baseband section 204) determines whether it needs to send or receive data over the message serial interface 208 (step 510). If so, the baseband section 204 determines if the message is a fast write message (step 512). If the message is a fast write message, then the baseband section 204 (if it is transmitting data) or the RF section 202 (if it is transmitting data) serially places 8 data bits on the appropriate serial message data line. Each data bit is shifted in by a message clock 404 transition for each data bit. (step 514). Otherwise, the baseband section 204 or the RF section 202 serially places all the data bits (e.g., 32 or 56 data bits) on the appropriate serial message line, with each data bit accompanied by a message clock 404 transition (step 516).

[0611] The slave select signal line may be used to transfer timing between the baseband section 204 and the RF section 202. In particular, the COUNT[19:0] output (see Table 6, Out_Dat = 8) represents the value of a counter present in the RF section 202 that reveals the sampling phase of the dual slope A/D converter in the RF section 202. The slave select signal line is connected to circuitry in the RF section 202 that latches DSP timing. Thus, the COUNT value is the value in the counter at the time the slave select signal causes the message that requests the COUNT output to latch the COUNT value into the shift register for transmission. In the baseband section 204, the slave select signal latches a counter (or another representation of time) when the slave selected signal is de-asserted (which is also when the RF section 202 latches COUNT).

[0612] The A/D sample timing may thereby be related to the baseband section 204 timing. The SHIFT[19:0] input (see Table 8, Address = 6) is used to shift the A/D timing to a desired offset from the baseband section 204 timing. As a result, the baseband section 204 may change the timing of the RF section 202 circuitry without additional interface lines.

[0613] Thus, systems and methods consistent with the invention provide power control messaging (and methods of operating or providing interfaces) between an RF processing section 202 and a baseband processing section 204. The messaging may be employed for many different purposes, and is particularly useful as part of general power control in an SPS device to reduce average power consumption and extend power supply life.

[0614] Typically, powering down as much of the RF section 202 as possible except when taking SPS signal samples helps reduce average power consumption. Taking the samples may in some instances occupy a time span as short as 10-20 ms in strong signal environments outdoors, or 50-100 ms in less favorable conditions outdoors. Indoors, the RF section 202 may operate for a time span on the order of a few seconds to obtain SPS signal samples, particularly for when the signal is weak. Note also that powering down the RF oscillator 212 when the baseband section 204 enters its own power down mode may also reduce power consumption.

[0615] More specifically, an exemplary operational sequence, including power control may proceed as shown below in Table 12:

Table 12	
Operational Step	Description
Initial Powerup	An alarm, timer, or wakeup circuit connected to or incorporated

	into the RF section 202 or baseband section 204 turns on a power supply connected to the RF section 202 and the baseband section 204.
Baseband Start	The RF oscillator 212 powers up and provides a clock signal to the baseband section 204. The baseband section 204 boots up using the clock signal.
Baseband Initialization	The baseband section 204 performs housekeeping tasks, Input / Output initialization, or other processing in preparation for RF section 202 startup.
RF Synthesizer Startup	The baseband section 204 powers up the RF clock synthesizer in the RF section 202 and waits a pre-determined time for the RF clock synthesizer to stabilize.
RF Circuitry Startup	The baseband section 204 turns on power to the LNA, AGC, A/D, and other selected circuitry in the RF section 202 and waits a pre-determined time for those sections to stabilize.
Sampling	The baseband section 204 starts taking data samples from the RF section 202.
Storage	In some modes of operation, the baseband section 204 directs storage of a block of data samples obtained from the RF section 202.
RF Circuitry Shutdown	The baseband section 204 turns off the LNA, AGC, A/D and RF clock synthesizer circuitry in the RF section 202.
GPS Measurement	The baseband section 204 directs processing of the stored data samples in order to extract GPS measurements from the data samples.
Location Update	The baseband section 204 determines a position update and delivers the update to a recipient over an Input / Output interface.
Wakeup Programming	The baseband section 204 programs the timer, alarm, or wakeup circuit for the next wake up alarm and begins shutdown.
Shutdown	The baseband section 204 initiates shutdown, gates off clocks, and

	powers down the RF section 202 and the baseband section 204 (except for the wakeup alarm circuitry).
--	--

[0616] The foregoing description of the preferred implementations of the invention has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed. Many modifications and variations are possible in light of the above teaching. It is intended that the scope of the invention not be limited by this detailed description.

APPENDIX B

Memory Sharing Among Components of Electronic Systems

Inventors:

Nicolas Vantalon
Steven A. Gronemeyer
Voya Protic

TECHNICAL FIELD

The disclosed embodiments relate to memory sharing among numerous components of electronic systems.

BRIEF DESCRIPTION OF THE FIGURES

Figure 1 Appendix B is a block diagram showing core memory sharing among components of an electronic system, under an embodiment.

Figure 2 Appendix B is a block diagram showing no memory sharing between a central processor and a digital signal processor (DSP) of an electronic system, under the embodiment of Figure 1.

Figure 3 Appendix B is a block diagram showing memory sharing between a central processor and the DSP of an electronic system, under the embodiment of Figure 1.

Figure 4 Appendix B is a block diagram showing memory sharing between a central processor and the DSP of an electronic system, under an alternative embodiment of Figure 3.

Figure 5 Appendix B is a block diagram showing an RTL implementation of memory sharing between a central processor and the DSP of an electronic system, under an alternative embodiment of Figure 4.

Figure 6 Appendix B is a block diagram of a memory filling order, under the memory sharing of an embodiment.

Figures 7A-7D Appendix B are block diagrams of memory mapping that supports memory sharing, under an embodiment.

Figure 8 Appendix B is a block diagram of a control register for memory sharing, under an embodiment.

Figure 9 Appendix B is a block diagram of a status register for memory sharing, under an embodiment.

Figure 10 Appendix B is a block diagram of a read/write violation address register for memory sharing, under an embodiment.

Figures 11A and 11B Appendix B show an address map for memory sharing, under an embodiment.

Figure 12 Appendix B is a block diagram showing core memory sharing among components of an electronic system, under an alternative embodiment of Figure 1.

Figure 13 Appendix B is a block diagram of the FS-RAMMOD module of an electronic system, under the embodiment of Figure 12.

Figure 14 Appendix B is a block diagram of random access memory (RAM) mapping and connection in the DSP of an electronic system, under the embodiment of Figure 12.

Figures 15A and 15B Appendix B show an input/output table, under the embodiment of Figure 12.

In the drawings, the same reference numbers identify identical or substantially similar elements or acts. To easily identify the discussion of any particular element or act, the most significant digit or digits in a reference number refer to the Figure number in which that element is first introduced (e.g., element 124 is first introduced and discussed with respect to Figure 1).

DETAILED DESCRIPTION

Devices and methods for sharing memory among components of an electronic system are provided herein. Memory locations of a memory device can be shared by a first processor, for example a central processor or microprocessor, and a second processor, for example a digital signal processor (DSP). The memory sharing includes providing the microprocessor indirect access to the shared memory via the DSP and an associated bus. The memory sharing further includes providing the microprocessor direct access to the shared memory through an associated bus. The memory sharing also includes partitioning the shared memory to simultaneously provide the microprocessor both the indirect and direct access to the shared memory.

In the following description, numerous specific details are introduced to provide a thorough understanding of, and enabling description for, embodiments of the memory sharing. One skilled in the relevant art, however, will recognize that the memory sharing can be practiced without one or more of the specific details, or with other components, systems, etc. In other instances, well-known structures or operations are not shown, or are not described in detail, to avoid obscuring aspects of the memory sharing.

Figure 1 Appendix B is a block diagram showing core memory sharing among components of an electronic system 100, under an embodiment. As an example, the electronic system 100 includes a first processor 112 and a first memory device 114 coupled to at least one bus 102. In operation the first processor 112 accesses the first memory device 114 via the bus 102 in order to write to and read from the first memory device 114.

A second processor 122 is coupled to the bus 102 using a first bridge unit 132. In operation the second processor 122 directly accesses the second memory device 124 in order to write to and read from the second memory device 124. The second memory device 124, in addition to coupling to the second processor 122, couples to the bus 102 using a second bridge unit 134.

The processors 112 and 122 include any collection of computing components and devices operating together, as is known in the art. The processors 112 and 122 can also be components or subsystems within a larger computer system or network. The processors 112 and 122 can also be

coupled among any number of components (not shown) known in the art, for example other buses, controllers, memory devices, and data input/output (I/O) devices, in any number of combinations.

The bus 102 can include any medium by which files are communicated or transferred between the processing systems or components of processing systems. Therefore, the paths represented by the bus 102 include wireless connections, wired connections, and hybrid wireless/wired connections. The paths also include couplings or connections to networks including local area networks (LANs), metropolitan area networks (MANs), wide area networks (WANs), proprietary networks, interoffice or backend networks, and the Internet.

Further to the electronic system 100 is a system in which the first processor 112 includes a central processing unit (CPU) or microprocessor like, for example, an ARM microprocessor, alternatively referred to herein as an ARM or a microprocessor. The second processor 122 includes a digital signal processor (DSP), but is not so limited. The first 114 and second 124 memory devices of this example include random access memory (RAM), but can include numerous other types of memory devices known in the art. While the first and second memory devices are shown herein as two separate devices, alternative embodiments of the memory sharing described herein can include any number of memory locations distributed among any number/combination of memory devices.

The electronic system 100 of an embodiment shares memory of the second memory device 124 among the first processor 112 and the second processor 122 using a number of methods. One method of sharing memory allows the first processor 112 access to the second memory device 124 via the first bridge unit 132 and the second processor 122. Another method of sharing memory allows the first processor 112 to directly access the second memory device 124 via the second bridge unit 134.

Figure 2 Appendix B is a block diagram of an electronic system 200 that supports memory sharing between a microprocessor 212 and a digital signal processor (DSP) 222, under the embodiment of Figure 1. The electronic system 200 includes the microprocessor 212 and a first memory device 214 coupled to at least one bus 202. A second processor 222 includes a DSP (Waverider) 222 coupled to the bus 202 using a first bridge unit 232. In operation the second

processor 222 directly accesses the second memory device 224 in order to write to and read from the second memory device 224.

Figure 3 Appendix B is a block diagram showing memory sharing between a central processor and the DSP of an electronic system, under the embodiment of Figure 1.

Figure 4 Appendix B is a block diagram showing memory sharing between a central processor and the DSP of an electronic system, under an alternative embodiment of Figure 3. The decoder 242 of an embodiment controls memory sharing according to the following.

```
If ( 60000000 =< cpu Address < 60010000 )  
    then Dsel_onramA Active;  
If ( 60010000 =< cpu Address < 60020000 ) and no fast ram  
    then Dsel_onraB Active;  
If ( 60010000 =< cpu Address < 60018000 ) and fast ram  
    then Dsel_onraC Active; and  
If ( 60018000 =< cpu Address < 60020000 ) and fast ram  
    then Dsel_onraB Active.
```

Figure 5 Appendix B is a block diagram showing an RTL implementation of memory sharing between a central processor and the DSP of an electronic system, under an alternative embodiment of Figure 4.

Figure 6 Appendix B is a block diagram of a memory filling order, under the memory sharing of an embodiment.

Figures 7A-7D Appendix B are block diagrams of memory mapping that supports memory sharing, under an embodiment. While four examples of memory sharing in this electronic system 100 including a microprocessor and DSP follow, but many alternative memory sharing schemes can be realized within the scope of the description herein.

Figure 7A Appendix B is a block diagram of a memory sharing configuration following electronic system start. At system start, also referred to as boot-up, the ARM has 64 kilobytes (kbytes) of fast RAM on its ASB. The DSP has 128 kbytes of RAM on its bus. The ARM sees its RAM at a base of 0x6000_0000. It can also see the DSP RAM through the DSP Bridge, SBU2. The ARM sees the DSP RAM at a base address of 0xC020_0000. The ARM access of DSP RAM

through the asynchronous bridge is slow. The ARM is allowed to read and write in the DSP address space. This allows the ARM to set up channel buffers and access report buffers.

If the ARM requires an additional amount of memory for ARM data, it can reallocate some of the DSP RAM for this purpose. Two methods are available: (1) switching RAM from the DSP bus to the ARM bus (ASB) and (2) memory mapping through the bridge. Both methods use memory mapping to place the reallocated RAM into contiguous ARM address space. Both methods allow for memory protection from unintended reads and writes. The switching method allows for fast RAM access, but only allows for switching a single 32k byte block. The memory mapping through the bridge allows for allocating up to eight smaller incremental 8k byte blocks, but provides only slow access. The two methods may be used in combination.

Figure 7B Appendix B is a block diagram of a memory sharing configuration in which a first segment of memory is mapped from the DSP to the microprocessor address space through a bridge unit. This mapping example shows the allocation of the first slow 8k-block. The block appears to the ARM as contiguous memory just above its initial RAM block. The memory lost by the DSP is removed from the top of DSP memory, so that the remaining memory is a contiguous block. If memory protection features are enabled, the address boundaries for this behavior are adjusted as shown. An abort is generated if the ARM accesses above the added memory block. The DSP generates an interrupt flag if the DSP accesses RAM above its reduced limit. The abort and interrupt address limits are shown as arrows in the figure.

Figure 7C Appendix B is a block diagram of a memory sharing configuration in which a number n of memory segments are mapped from the DSP to the microprocessor address space through a bridge unit. This mapping example shows how additional 8k-blocks are mapped from the DSP address space to the ARM address space through the bridge. Blocks are mapped in order so that memory is removed from the top of DSP RAM and added to the top of ARM RAM. Thus, the DSP address-space is reduced from top to bottom, while the address-space of the ARM grows from bottom to top. The abort and interrupt points are adjusted accordingly.

Figure 7D Appendix B is a block diagram of a memory sharing configuration in which a portion of the memory is shared by switching access to the memory from the DSP bus to the ARM

bus, and a portion of the memory is shared via mapping from the DSP to the ARM address space through the bridge, under an embodiment. This mapping example shows how the switched block is mapped. The switched block contains the first four mapped blocks. If the ARM had previously mapped one or more of these blocks as slow ARM memory, this data remains intact at the same addresses when the block is later mapped as fast memory. The ARM may have already mapped one or more of the other four blocks before the switch or else it may latter map one or more of these blocks after the switch. Existing data ordering is preserved and the switching method takes precedence over the bridge mapping method.

Additional attributes of the implementation are as follows:

1. A register field is provided to map from one to eight blocks of 8k bytes each from DSP address space to ARM address space. The blocks are mapped in order from the top of DSP memory. Access to this RAM is via the asynchronous bridge.
2. A register field is provided to switch the top 32k of DSP RAM from the DSP bus to the ARM bus (ASB). If one or more of the corresponding four 8k blocks of RAM were already or are subsequently mapped, the switch control takes precedence. Data contents of the first four mapped blocks of 8k and the single switched 32k block are at identical offsets and byte ordering with either reallocation method.
3. A register field is provided to enable or disable the generation of aborts for CPU DSP RAM read or write access violations. Generation of these events can also be independently enabled or disabled for read and write accesses.
4. A register is provided to save the address causing the initial DSP memory violation and subsequent interrupt.
5. Access to all ARM and DSP RAM allows byte, half-word and word access with byte lane write controls. This attribute holds both for switched RAM and for RAM mapped through the bridge. Byte ordering for the DSP RAM is identical for either the DSP base-address or the mapped or switched blocks using the ARM base-address.
6. Access to non-RAM DSP core registers is via SBU2. The address mapping for these registers is relative to the SBU2 address space.

Figure 8 Appendix B is a block diagram of a control register for memory sharing, under an embodiment. A description of this register is as follows:

- 15:7 Reserved
- 6 DSP32K_SWI_ENB; 1= enable CPU direct access of the upper 32Kbyte of DSP RAM through the ASB.
- 5:3 MAP_BLK[2:0]; 000-111 selects from 1 to 8 8Kbyte blocks of the upper 64K bytes of DSP RAM for CPU soft mapping access when DSP64K_MAP_ENB=1. It set the CPU soft MAP address boundary as follows:
 - 000: $\geq 0x6001_0000 < 0x6001_2000$ or $\geq 0xC021_E000 < 0xC022_0000$
 - 001: $\geq 0x6001_0000 < 0x6001_4000$ or $\geq 0xC021_C000 < 0xC022_0000$
 - 010: $\geq 0x6001_0000 < 0x6001_6000$ or $\geq 0xC021_A000 < 0xC022_0000$
 - 011: $\geq 0x6001_0000 < 0x6001_8000$ or $\geq 0xC021_8000 < 0xC022_0000$
 - 100: $\geq 0x6001_0000 < 0x6001_A000$ or $\geq 0xC021_6000 < 0xC022_0000$
 - 101: $\geq 0x6001_0000 < 0x6001_C000$ or $\geq 0xC021_4000 < 0xC022_0000$
 - 110: $\geq 0x6001_0000 < 0x6001_E000$ or $\geq 0xC021_2000 < 0xC022_0000$
 - 111: $\geq 0x6001_0000 < 0x6002_0000$ or $\geq 0xC021_0000 < 0xC022_0000$
- 2 DSP64K_MAP_ENB; 1= enable CPU soft mapping from 1 to eight 8Kbyte blocks of the upper 64K bytes of DSP RAM at address $\geq 0x6001_0000 < 0x6002_0000$ through the SBU2 bus bridge.
- 1 EN_CPU_WAB; 1 = enable abort on μ P write violation.
- 0 EN_CPU_RAB; 1 = enable abort on μ P read violation.

A more detailed explanation of each control bit follows. The EN_CPU_RAB bit enables the microprocessor (μ P) to abort the access if it attempts to read from DSP RAM in an address range that is not allowed (NA) because it is mapped for μ P use or not physically present on the bus being accessed. For example, if SWI_ENB = 0 and MAP_ENB = 1 and the μ P tries to read from the address range 0x6001_C000 to 0x6001_FFFF, an abort is generated. Similarly, if SWI_ENB = 1 and the μ P tries to read from the range 0xC021_E000 to 0xC021_FFFF, an abort is generated. Note, however, that if SWI_ENB = 0 and MAP_ENB = 1 and MAP_BLK \geq 000, the μ P may read from block 0 through the bridge at both address ranges 0x6001_0000 to 0x6001_1FFF and 0xC021_E000 to 0xC021_FFFF.

The EN_CPU_WAB bit enables the microprocessor (μ P) to abort the access if it attempts to write to DSP RAM in an address range that is not allowed (NA) because it is mapped for μ P use or

not physically present on the bus being accessed. For example, if SWI_ENB = 0 and MAP_ENB = 1 and the μ P tries to write to the address range 0x6001_C000 to 0x6001_FFFF, an abort is generated. Similarly, if SWI_ENB = 1 and the μ P tries to write to the range 0xC021_E000 to 0xC021_FFFF, an abort is generated because the block is not present on the DSP bus. Note, however, that if SWI_ENB = 0 and MAP_ENB = 1 and MAP_BLK \geq 000, the μ P may write to block 0 through the bridge at both address ranges 0x6001_0000 to 0x6001_1FFF and 0xC021_E000 to 0xC021_FFFF.

The DSP64K_MAP_ENB bit enables MAP_BLK [2:0] to control the address mapping of eight 8k-byte blocks from the DSP address range to the address range directly above the on-chip ARM RAM. When this bit is disabled (set to zero), the mapping bits MAP_BLK [2:0] have no effect.

The MAP_BLK [2:0] bit controls the mapping of eight 8k-byte blocks from the DSP address range to the address range directly above the on-chip ARM RAM. The bytes are always mapped beginning with block zero and ending with the block specified by MAP_BLK [2:0]. Block 0 is at the high end of the DSP address range and is mapped to the low end of the mapped range above ARM RAM. This "reverse stacking" is done so that the remaining DSP RAM is a contiguous address range as upper blocks are removed and the added ARM RAM grows upward with each added block as a contiguous address range.

The DSP32K_SWI_ENB bit switches blocks 0 to 3 from the DSP bus to the ARM ASB. The four blocks are stacked in reverse order in the ASB address range, beginning with block 0 being mapped just above the top of on-chip ARM RAM. This same mapping is used when MAP_BLK [2:0] maps the blocks via SBU2.

Figure 9 Appendix B is a block diagram of a status register for memory sharing, under an embodiment. A description of this register is as follows:

- 15:2 Reserved
- 1 CPUW_VIO. For read, 1= \Rightarrow μ P write-violation occurs. For write, writing a 1 will clear the bit.

0 CPUR_VIO. For read, 1=> μ P read-violation occurs. For write, writing a 1 will clear the bit.

A more detailed explanation of each control bit follows. The CPUR_VIO bit is set when μ P read-violations occurs. The bit is set independently of the state of EN_CPU_RAB. The bit is cleared when the μ P writes back a one. The CPUW_VIO bit is set when μ P write-violations occurs. The bit is set independently of the state of EN_CPU_WAB. The bit is cleared when the μ P writes back a one.

Figure 10 Appendix B is a block diagram of a read/write violation address register for memory sharing, under an embodiment. A description of this register is as follows:

15:0 DSP address first causing DSPW_INT or DSPR_INT pulse.

Figures 11A and 11B Appendix B show an address map for memory sharing, under an embodiment. Some examples are presented below of the address mapping for memory sharing, but the embodiment is not so limited.

In a first example, Block = 7, SWI_ENB = X, MAP_ENB = 0, MAP_BLK [2:0] = XXX. The DSP accesses the block in its normal DSP address range 0x0001_0000 to 0x0001_FFFF. The ARM accesses the block at its normal SBU2 address of 0xC021_0000 to 0xC021_1FFF. The ARM must not access this block on the ASB.

In a second example, Block = 7, SWI_ENB = X, MAP_ENB = 1, MAP_BLK [2:0] = 110. The DSP accesses the block in its normal DSP address range 0x0001_0000 to 0x0001_FFFF. The ARM accesses the block at its normal SBU2 address of 0xC021_0000 to 0xC021_FFFF or the soft map address 0x6001_0000 to 0x6001_5FFF. The ARM must not access this block on the ASB.

In a third example, Block = 7, SWI_ENB = X, MAP_ENB = 1, MAP_BLK [2:0] = 111. The DSP is not programmed to access the block in its normal DSP address range 0x0001_0000 to 0x0001_FFFF. The ARM accesses the block at its mapped SBU2 address of 0x6001_0000 to 0x6001_7FFF. The ARM does not access this block on the ASB. If the DSP tries to access this block in its normal DSP bus address range, DSP read or write interrupts should be generated.

Figure 12 Appendix B is a block diagram showing core memory sharing among components of an electronic system, under an alternative embodiment of Figure 1.

Figure 13 Appendix B is a block diagram of the FS-RAMMOD module of an electronic system, under the embodiment of Figure 12.

Figure 14 Appendix B is a block diagram of random access memory (RAM) mapping and connection in the DSP of an electronic system, under the embodiment of Figure 12.

Figures 15A and 15B Appendix B show an input/output table, under the embodiment of Figure 12.

Referring to **Figures 12, 13, 14, 15A, and 15B, Appendix B** the FS_RAMMOD module carries out the CPU direct access to the DSP upper 32K bytes (out of 128K bytes) RAM through the ASB as part of the DSP RAM sharing function, but the embodiment is not so limited. The DSP RAM sharing function basically allows CPU to either directly access (in CPU clock domain, BCLK) the DSP RAM on the fast ASB bus and/or indirectly access the DSP RAM (in DSP clock domain, SPCLK) through the slower DSP clock to CPU clock domain conversion. The DSP RAM is generally controlled by the DSP clock, SPCLK. The fast CPU direct access is accomplished by switching the upper 32K bytes DSP RAM clock to BCLK, the CPU clock.

The DSP RAM sharing function of an embodiment is partitioned to be carried out by several modules including the FS_RAMMOD module, the SBU2MOD bridge module, the ADDRMON module, the DECMOD module, the ITMOD and the DSP (Waverider) core.

The FS_RAMMOD module manages all DSP RAM sharing registers access, provides DSP RAM Sharing control signals to all other modules, collects DSP RAM sharing status signals from all other modules, and carries out CPU direct access to DSP upper 32K bytes (out of 128K bytes) RAM on the ASB.

The SBU2MOD module arbitrates between CPU soft mapping access and DSP core access to the DSP RAM, carries out CPU indirect access to DSP 128K bytes RAM at 0xC000_0000 range (no soft mapping of CPU addresses), carries out CPU indirect access to DSP upper 64K bytes RAM from 0x6001_0000 to 0x6001_FFFF address range through soft mapping, detects any CPU DSP RAM read or write access violation base on DSP RAM sharing control bits and generates violation status bits, and generates CPU abort cycle if enabled to do so during read or write access violation.

The ADDRMON module detects Waverider Core DSP RAM access violation base on DSP RAM sharing control bits and generates the DSP_RVIO_INT and DSP_WVIO_INT interrupt pulses to ITMOD, and captures Waverider Core DSP RAM access violation addresses and sends them to the FS_RAMMOD module.

Aspects of the memory sharing of an embodiment may be implemented as functionality programmed into any of a variety of circuitry, including programmable logic devices (PLDs), such as field programmable gate arrays (FPGAs), programmable array logic (PAL) devices, electrically programmable logic and memory devices and standard cell-based devices, as well as application specific integrated circuits (ASICs). Some other possibilities for implementing aspects of the memory sharing of an embodiment include: microcontrollers with memory (such as electronically erasable programmable read only memory (EEPROM)), embedded microprocessors, firmware, software, etc. Furthermore, aspects of the memory sharing of an embodiment may be embodied in microprocessors having software-based circuit emulation, discrete logic (sequential and combinatorial), custom devices, fuzzy (neural) logic, quantum devices, and hybrids of any of the above device types. Of course the underlying device technologies may be provided in a variety of component types, e.g., metal-oxide semiconductor field-effect transistor (MOSFET) technologies like complementary metal-oxide semiconductor (CMOS), bipolar technologies like emitter-coupled logic (ECL), polymer technologies (e.g., silicon-conjugated polymer and metal-conjugated polymer-metal structures), mixed analog and digital, etc.

Unless the context clearly requires otherwise, throughout the description and the claims, the words "comprise," "comprising," and the like are to be construed in an inclusive sense as opposed to an exclusive or exhaustive sense; that is to say, in a sense of "including, but not limited to." Words using the singular or plural number also include the plural or singular number respectively. Additionally, the words "herein," "hereunder," "above," "below," and words of similar import, when used herein, shall refer to this patent as a whole and not to any particular portions of this patent. When the word "or" is used in reference to a list of two or more items, that word covers all of the following interpretations of the word: any of the items in the list, all of the items in the list and any combination of the items in the list.

The above description of illustrated embodiments of the memory sharing is not intended to be exhaustive or to limit the invention to the precise form disclosed. While specific embodiments of, and examples for, the memory sharing are described herein for illustrative purposes, various equivalent modifications are possible within the scope of the invention, as those skilled in the relevant art will recognize. The teachings of the memory sharing provided herein can be applied to other electronic systems, not only for the electronic systems described above.

The elements and acts of the various embodiments described above can be combined to provide further embodiments. These and other changes can be made to the memory sharing of an embodiment in light of the above detailed description.

All of the above references and United States Patents and patent applications are incorporated herein by reference. Aspects of the embodiments described herein can be modified, if necessary, to employ the systems, functions and concepts of the various patents and applications described above to provide yet further embodiments of the memory sharing.

In general, in the following claims, the terms used should not be construed to limit the memory sharing to the specific embodiments disclosed in the specification and the claims, but should be construed to include all electronic systems that operate under the claims to provide memory sharing. Accordingly, the memory sharing is not limited by the disclosure, but instead the scope of the memory sharing is to be determined entirely by the claims.

While certain aspects of the memory sharing are presented below in certain claim forms, the inventors contemplate the various aspects of the memory sharing in any number of claim forms. For example, while only one aspect of the memory sharing is recited as embodied in computer-readable medium, other aspects may likewise be embodied in computer-readable medium. Accordingly, the inventors reserve the right to add additional claims after filing the application to pursue such additional claim forms for other aspects of the memory sharing.

CLAIMS

What is claimed is:

1. A radio frequency (RF) to baseband interface providing power control over an RF section that processes RF signals and that is coupled to a baseband section that processes baseband signals, the interface comprising:

a bi-directional message interface for communicating a power control message from the baseband section to the RF section; and

a data interface for communicating data from the RF section to the baseband section.
2. The interface of claim 1, where the power control message comprises a power control bit specifying a power state for pre-selected circuitry in the RF section.
3. The interface of claim 2, where the power state is one of a power-up state and a power-down state.
4. The interface of claim 1, where the power control message comprises a plurality of power control bits individually specifying power states for a plurality of pre-selected circuitry in the RF section.
5. The interface of claim 2, where the pre-selected circuitry is at least one of a frequency divider, oscillator, and amplifier.
6. The interface of claim 1, where the message interface is a serial message interface.

7. The interface of claim 1, where the message interface comprises a message-in signal line, a message-out signal line and a message clock signal line.
8. A method for controlling power in a radio frequency (RF) section that processes RF signals and that is coupled to a baseband section that processes baseband signals, the method comprising the steps of:
 - setting a power control bit in a power control message; and
 - communicating the power control message over a message interface from the baseband section to the RF section.
9. The method of claim 8, wherein the step of communicating comprises the step of serially communicating the power control message.
10. The method of claim 8, wherein the step of communicating comprises the step of serially communicating the power control message using a message-in signal line, a message-out signal line and a message clock signal line.
11. The method of claim 8, where the power control bit specifies a power state for pre-selected circuitry in the RF section.
12. The method of claim 11, where the power state is one of a power-up state and a power-down state.
13. The method of claim 8, where the step of setting comprises the step of setting a plurality of power control bits individually specifying power states for a plurality of pre-selected circuitry in the RF section.
14. An RF front end for a satellite positioning system receiver, the front end comprising:

an RF processing section comprising an RF input for receiving satellite positioning system signals; and

an RF to baseband interface coupled to the RF processing section, the interface comprising:

a bi-directional message interface for communicating messages between the RF processing section and a baseband processing section, including receiving a power control message from the baseband processing section; and

a data interface for communicating data from the RF processing section to the baseband processing section.

15. The RF front end of claim 14, wherein the message interface comprises:

a message clock line;

a message-in signal line and

a message-out signal line; and

wherein the message-out signal line carries an output bit stream representing the power control message.

16. The RF front end of claim 15, where the power control message comprises a power control bit specifying a power state for pre-selected circuitry in the RF section.

17. The RF front end of claim 16, where the power state is one of a power-up state and a power-down state.

18. The RF front end of claim 15, where the power control message comprises a plurality of power control bits individually specifying power states for a plurality of pre-selected circuitry in the RF section.

19. The RF front end of claim 15, where the pre-selected circuitry is at least one of a frequency divider, oscillator, and amplifier.

20. The RF front end of claim 15, where the data interface comprises a data clock signal line and a data bit signal line.

21. The RF front end of claim 20, where:

the data clock signal line carries a data clock comprising a rising edge and a falling edge;

the data bit signal line carries a data signal comprising a sign bit and a magnitude bit;

and

the first data bit is valid on the rising edge of the data clock and the second data bit is valid on the falling edge of the data clock.

22. A baseband back end for a satellite positioning system receiver, the back end comprising:

a baseband processing section comprising at least one address, data, and control line for communicating with a digital device; and

an RF to baseband interface coupled to the baseband processing section, the interface comprising:

a bi-directional message interface for communicating messages between an RF processing section and the baseband processing section, including communicating a power control message to the RF processing section; and

a data serial interface for communicating data from the RF processing section to the baseband processing section.

23. The baseband back end of claim 22, wherein the message serial interface comprises:

a message clock line;

a message-in signal line and

a message-out signal line; and

wherein the message-out signal line carries an output bit stream representing the power control message.

24. The baseband back end of claim 22, where the power control message comprises a power control bit specifying a power state for pre-selected circuitry in the RF processing section.

25. The baseband back end of claim 24, where the power state is one of a power-up state and a power-down state.

26. The baseband back end of claim 22, where the power control message comprises a plurality of power control bits individually specifying power states for a plurality of pre-selected circuitry in the RF section.

27. The baseband back end of claim 26, where the pre-selected circuitry is at least one of a frequency divider, oscillator, and amplifier.

28. A satellite positioning system receiver comprising:

an RF front end comprising an RF processing section and an RF input for receiving satellite positioning system signals;

a baseband back end comprising a baseband processing section and at least one address, data, and control line for communicating with a digital device; and

an RF to baseband interface coupled between the RF processing section and the baseband processing section, the interface comprising:

a bi-directional message interface for communicating messages between the RF processing section and the baseband processing section, including communicating a power control message to the RF processing section; and

a data interface for communicating data from the RF processing section to the baseband processing section.

29. The satellite positioning system receiver of claim 28, wherein the message interface comprises:

a message clock line;

a message-in signal line and

a message-out signal line; and

wherein the message-out signal line carries an output bit stream representing the power control message.

30. The satellite positioning system receiver of claim 29, where the power control message comprises a power control bit specifying a power state for pre-selected circuitry in the RF processing section.

31. The satellite positioning system receiver of claim 30, where the power state is one of a power-up state and a power-down state.
32. The satellite positioning system receiver of claim 29, where the power control message comprises a plurality of power control bits individually specifying power states for a plurality of pre-selected circuitry in the RF section.
33. The satellite positioning system receiver of claim 32, where the pre-selected circuitry is at least one of a frequency divider, oscillator, and amplifier.
34. A satellite positioning subsystem comprising:
circuitry for implementing a receiver manager expert system.
35. A satellite positioning subsystem comprising:
circuitry for implementing joint detection carrier centering bit synchronization acquisition.
36. A satellite positioning subsystem comprising:
circuitry for implementing zoom detection and interpolation.
37. A satellite positioning subsystem comprising:
circuitry for implementing multi-dimensional measurement interpolation.
38. A satellite positioning subsystem comprising:
circuitry for implementing mode switching between a navigation signal without synchronization and with synchronization.
39. A satellite positioning subsystem comprising circuitry for implementing receiver autonomous integrity monitoring.

U.S. Express Mail No.: EUUS
Filing Date: August 31, 2003

PATENT
Docket No. ST02042USV (281-US-P1)

ABSTRACT OF THE DISCLOSURE

Control and feature systems for processing signals from a satellite positioning system include an Expert System receiver manager; a joint detection, carrier centering and bit sync acquisition subsystem; a zoom in, zoom out detection and interpolation subsystem; a multi-dimensional measurement interpolation subsystem; a subsystem for mode switching between a navigational signal with Synch and without Synch; and an autonomous integrity monitoring subsystem for a receiver.

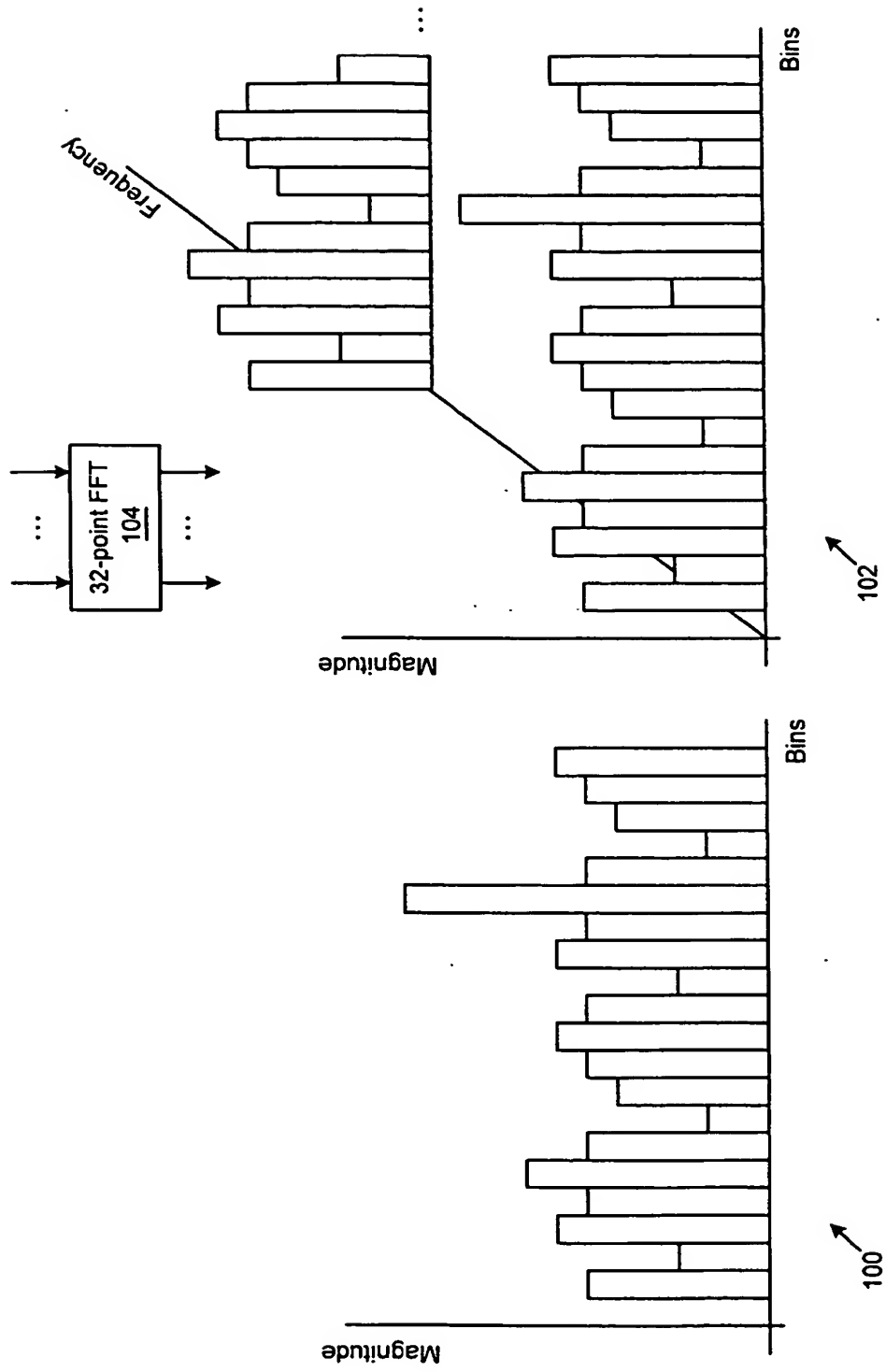


Figure 1

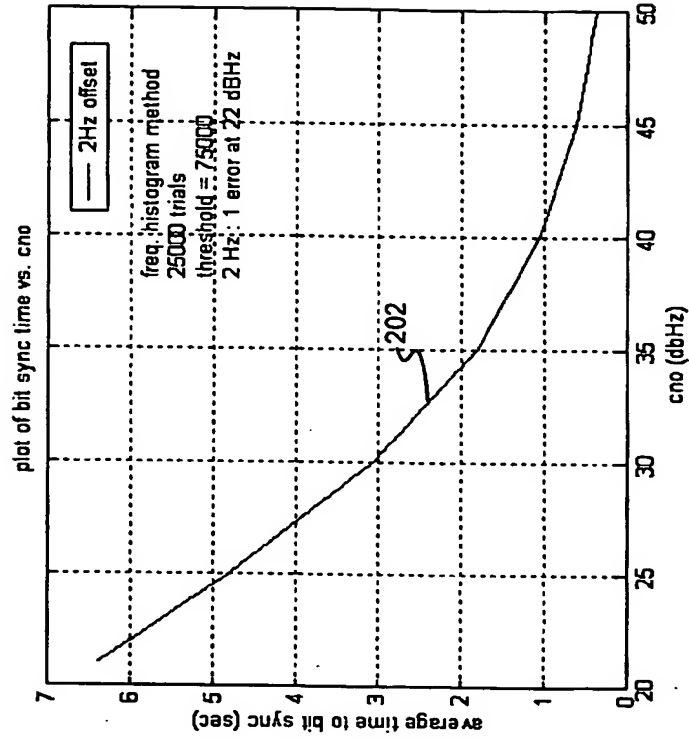
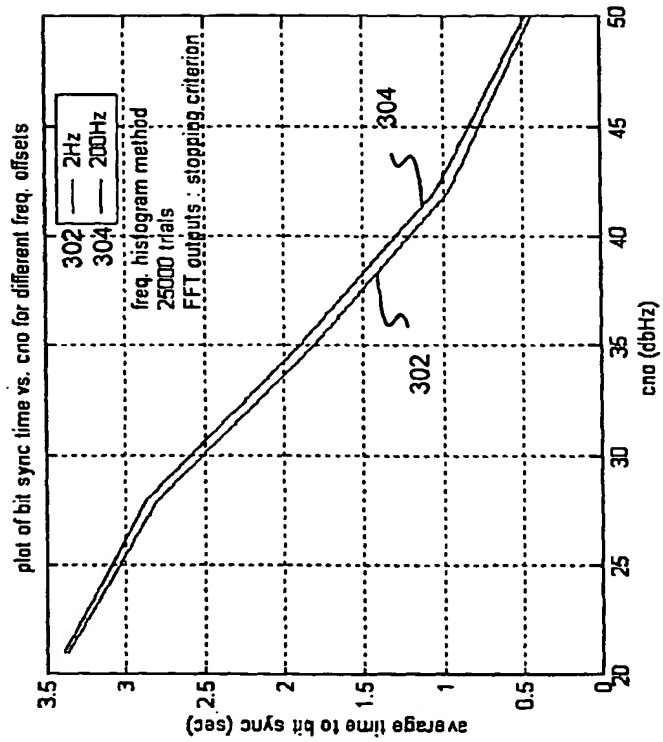


Figure 2



300

Figure 3

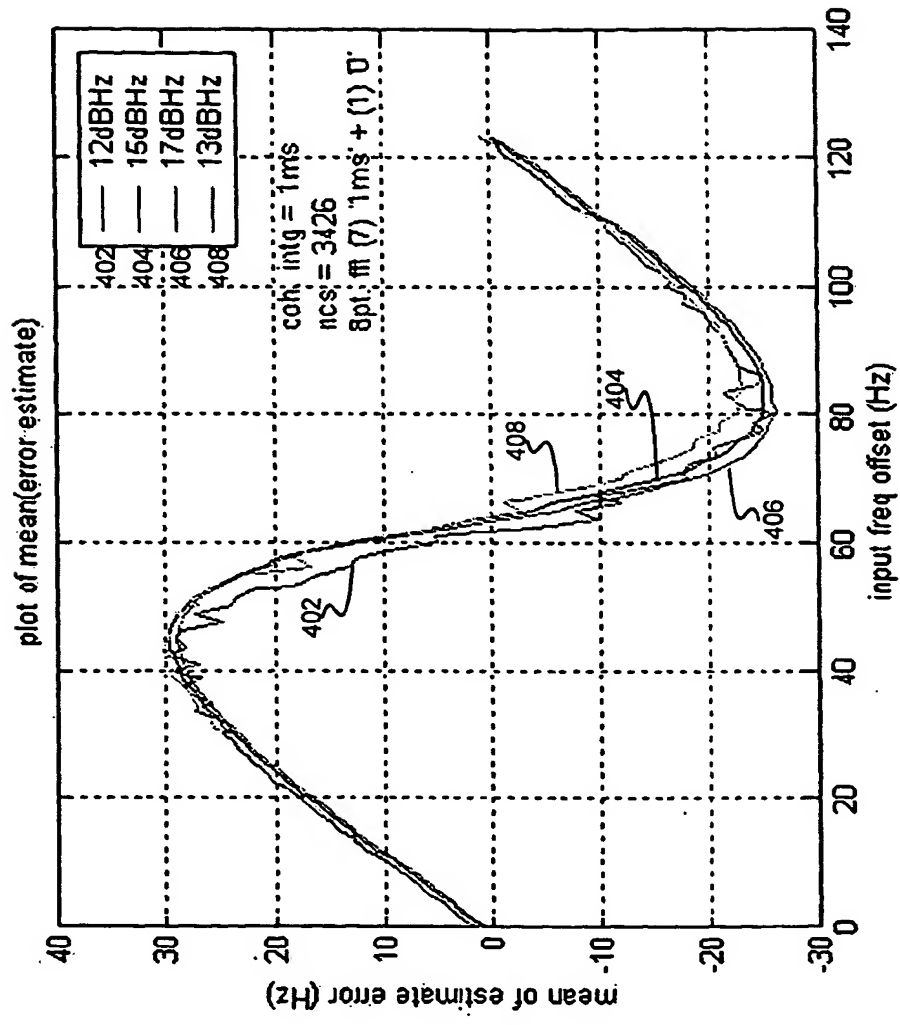


Figure 4

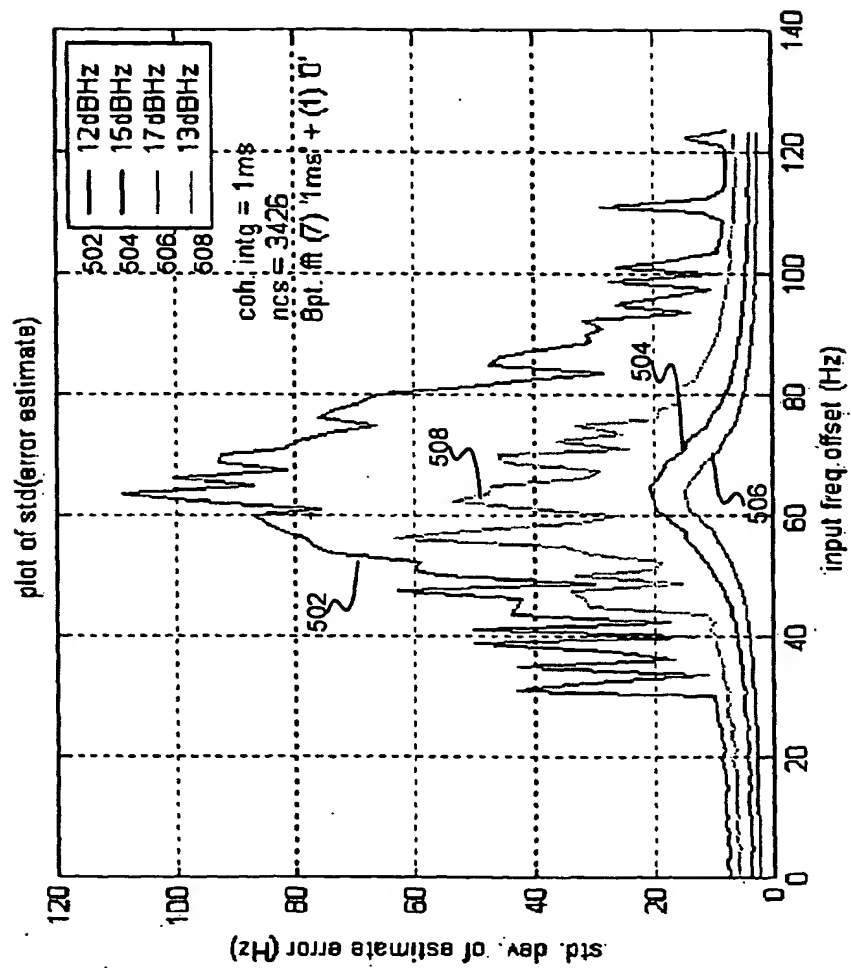


Figure 5

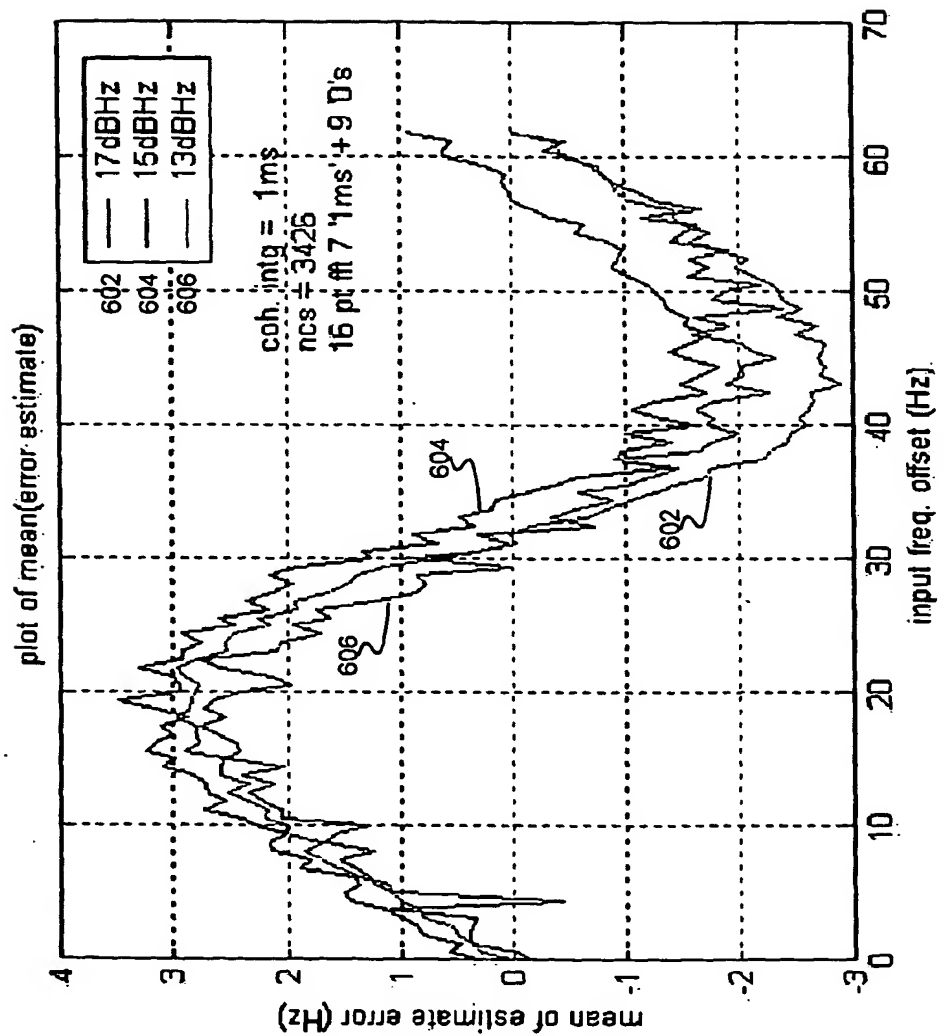


Figure 6

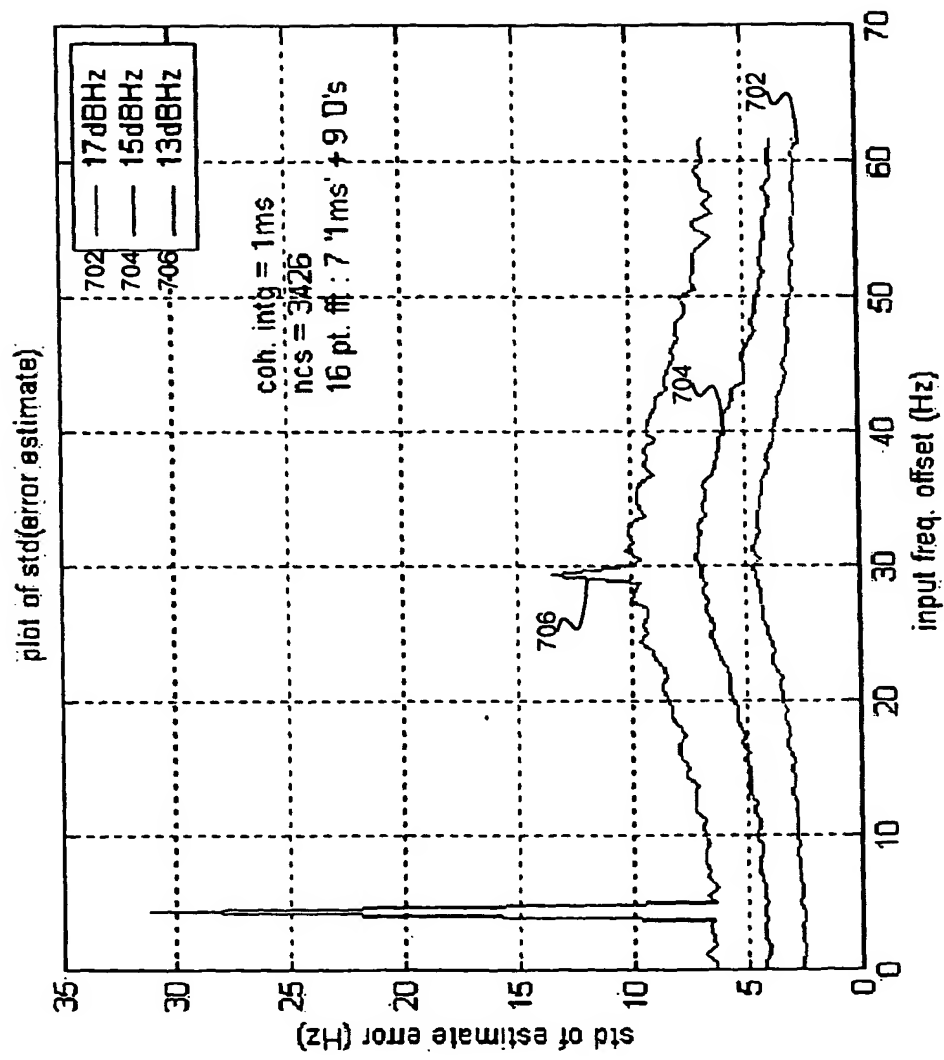


Figure 7

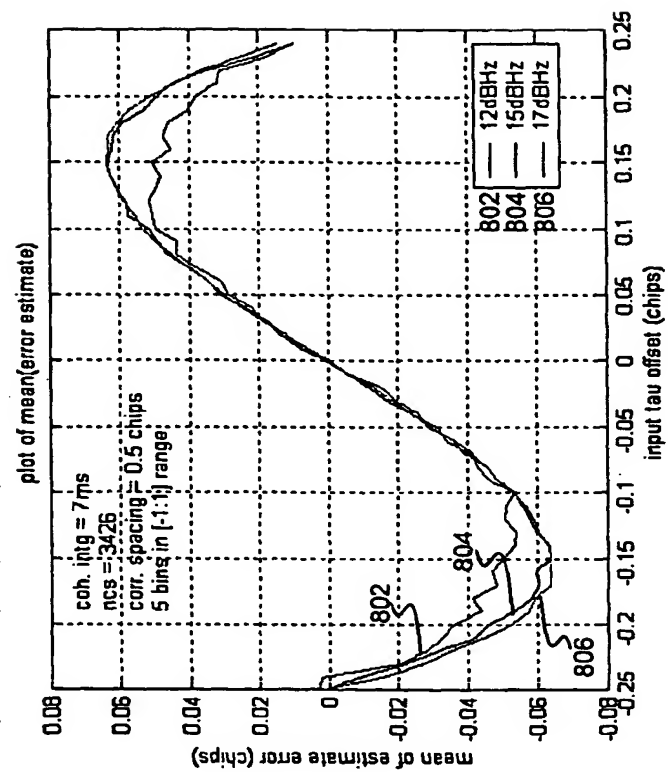


Figure 8

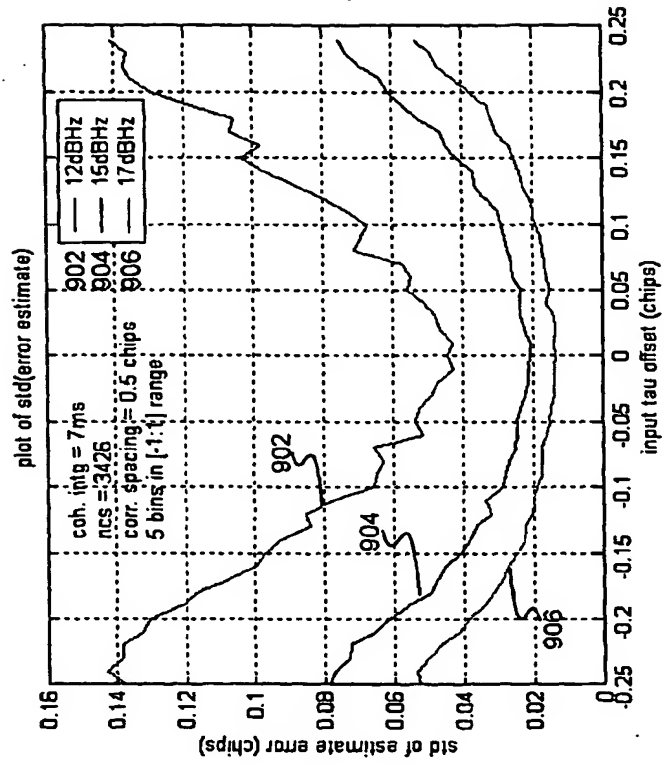


Figure 9

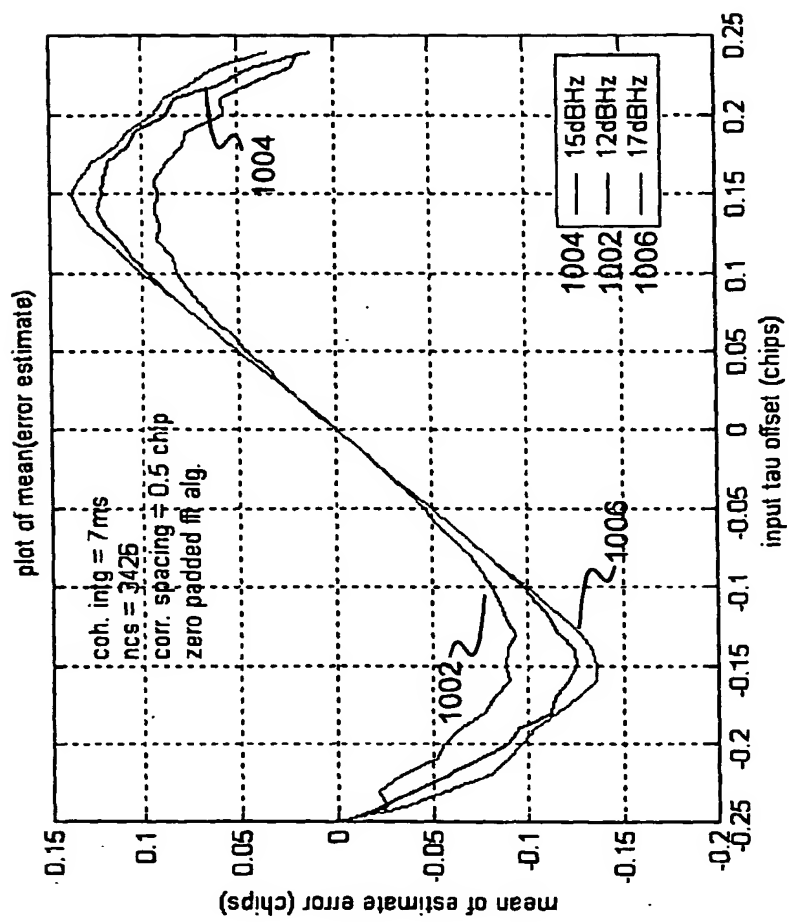


Figure 10

1000

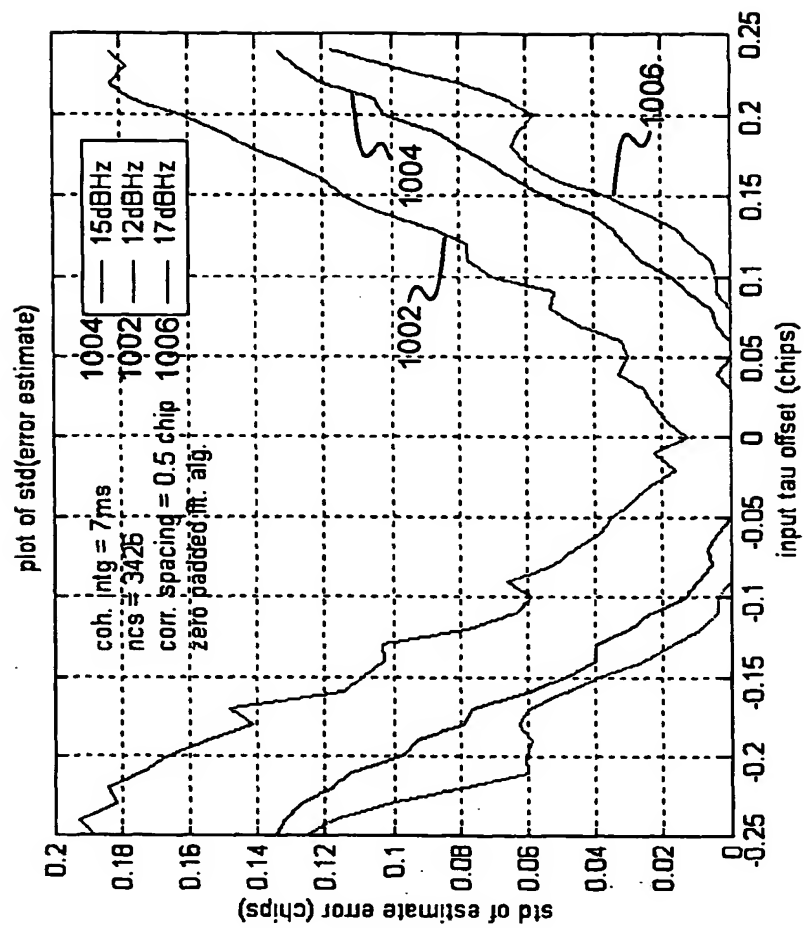


Figure 11

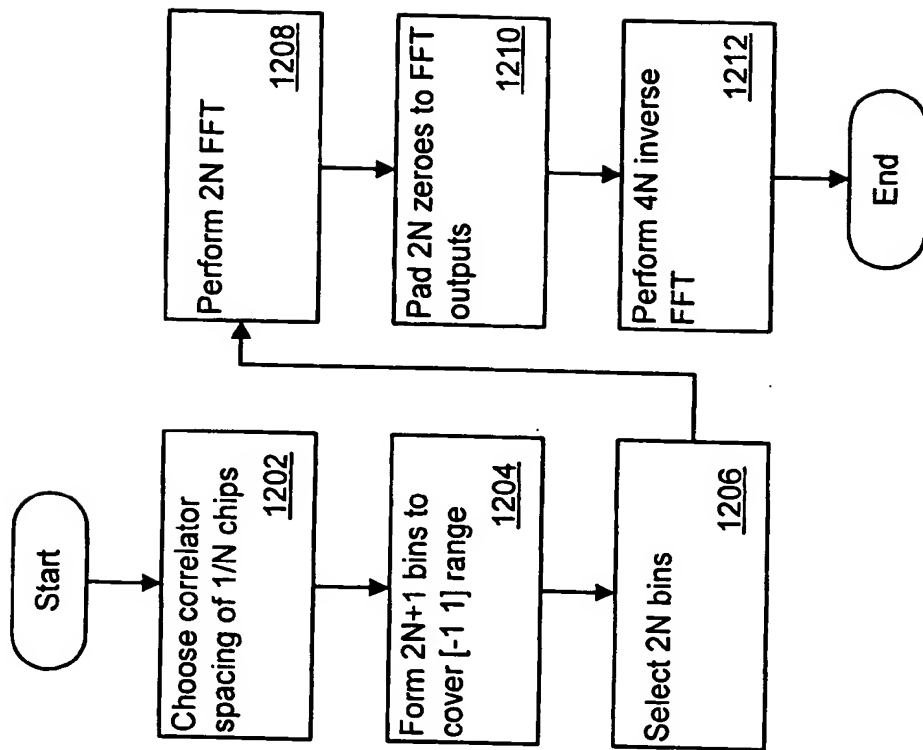


Figure 12

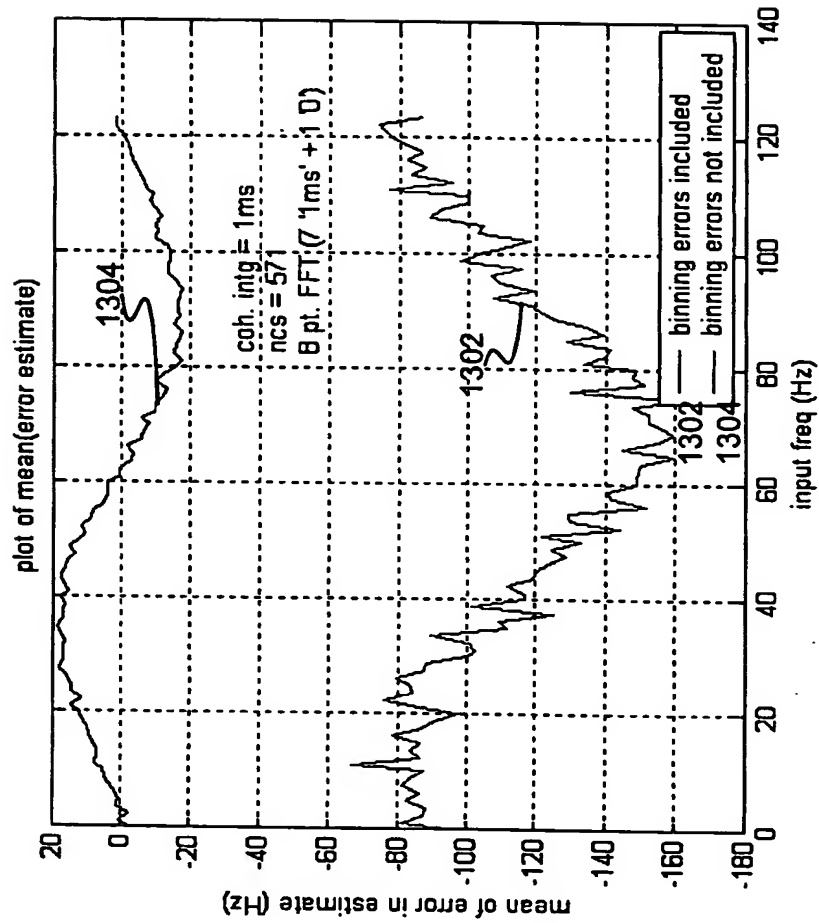


Figure 13

1300

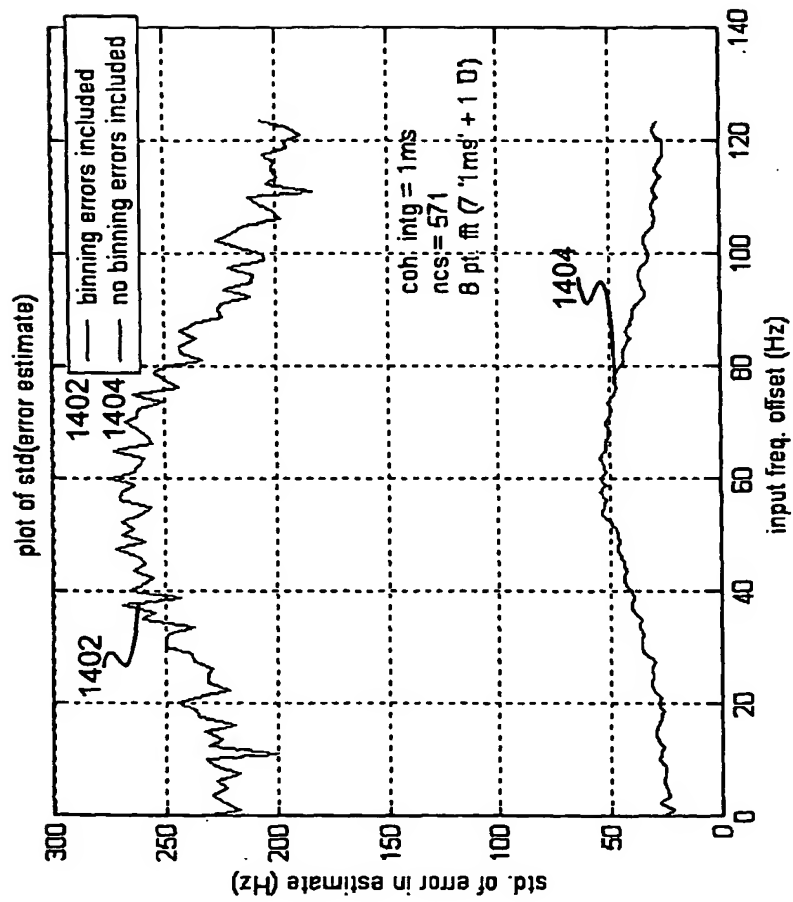
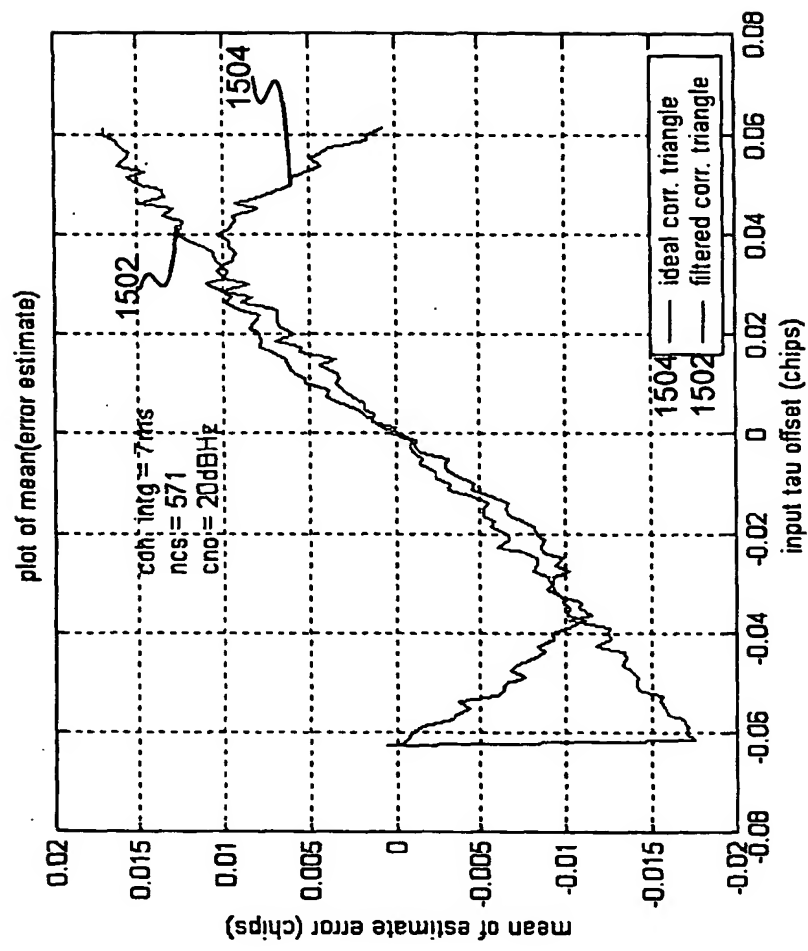


Figure 14

1400



1500

Figure 15

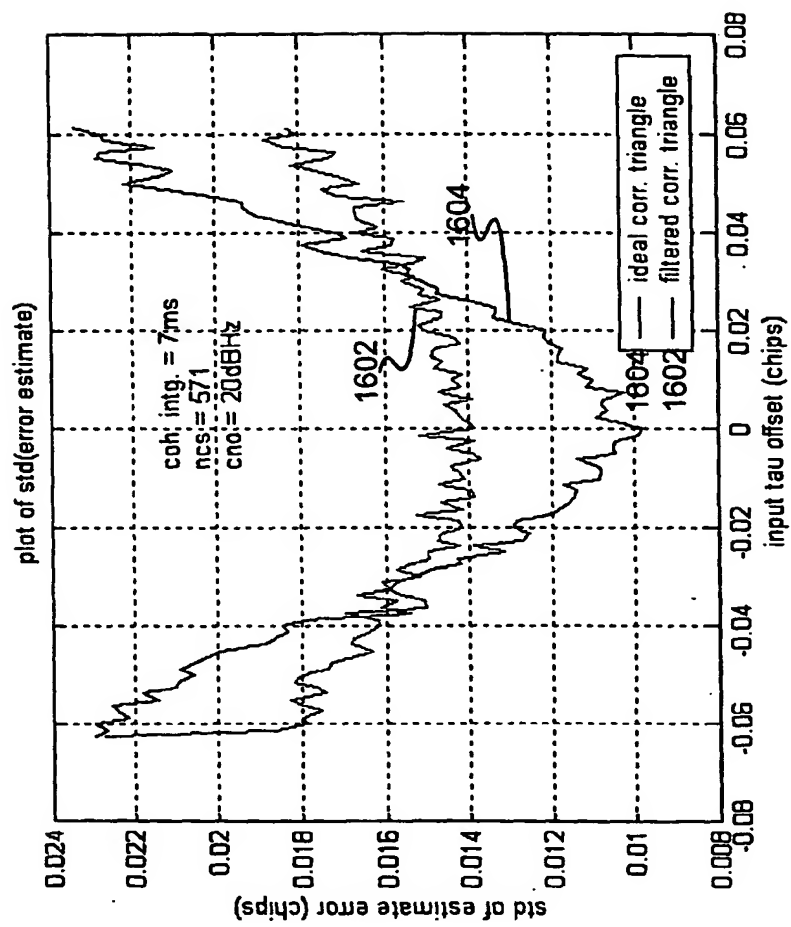
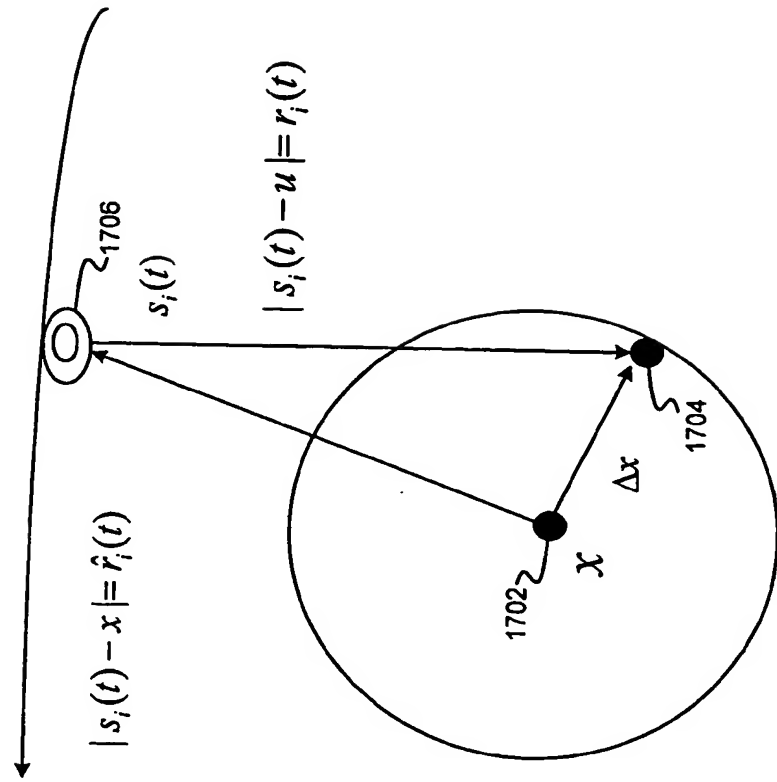


Figure 16



1700

Figure 17

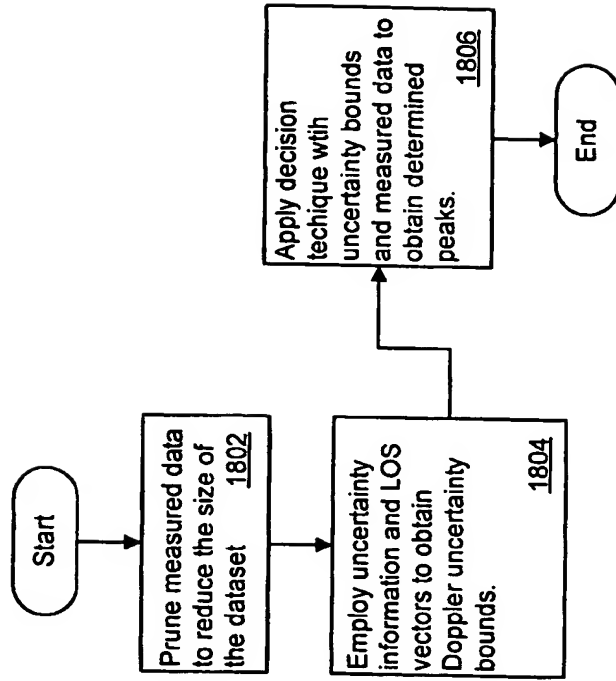


Figure 18

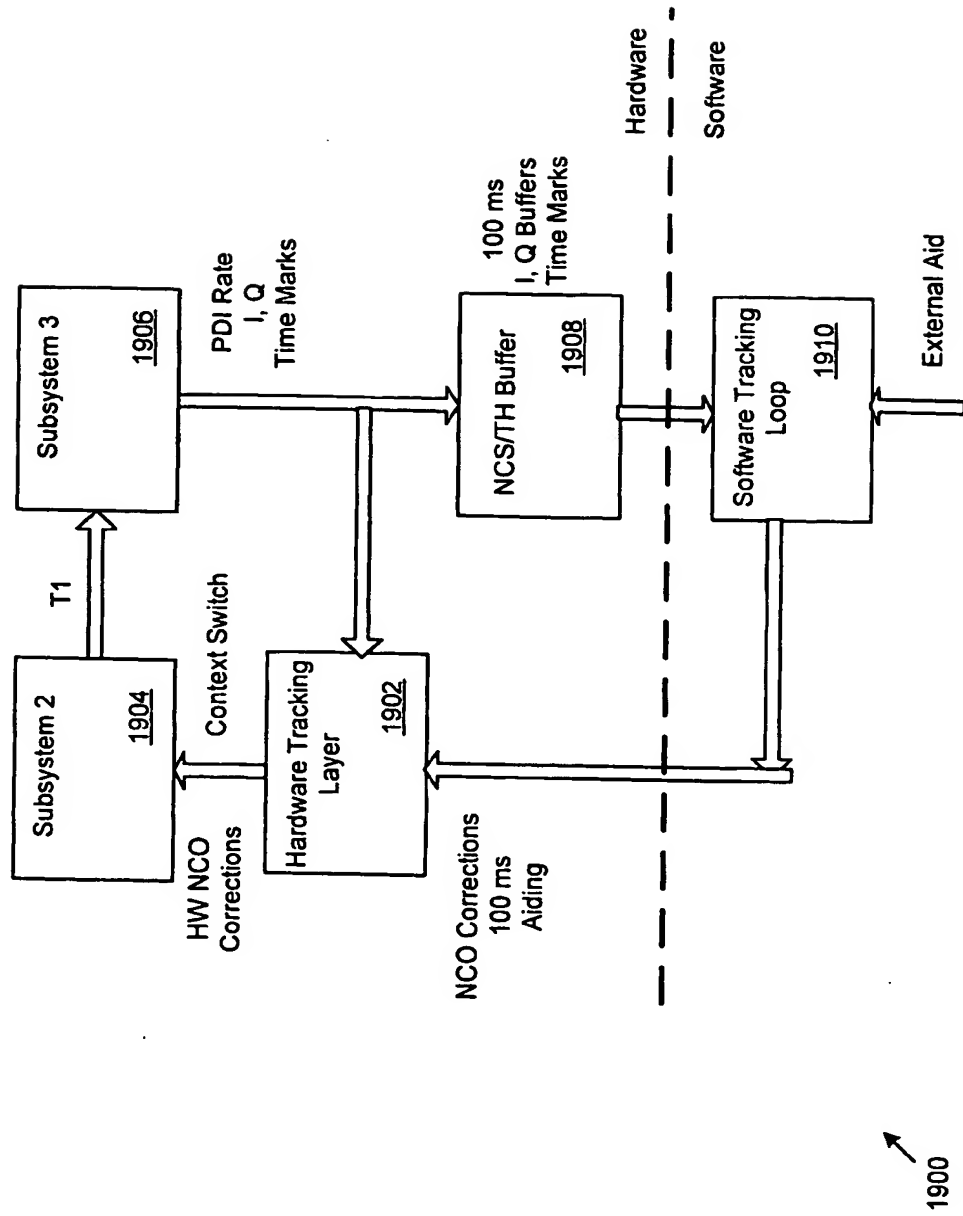


Figure 19

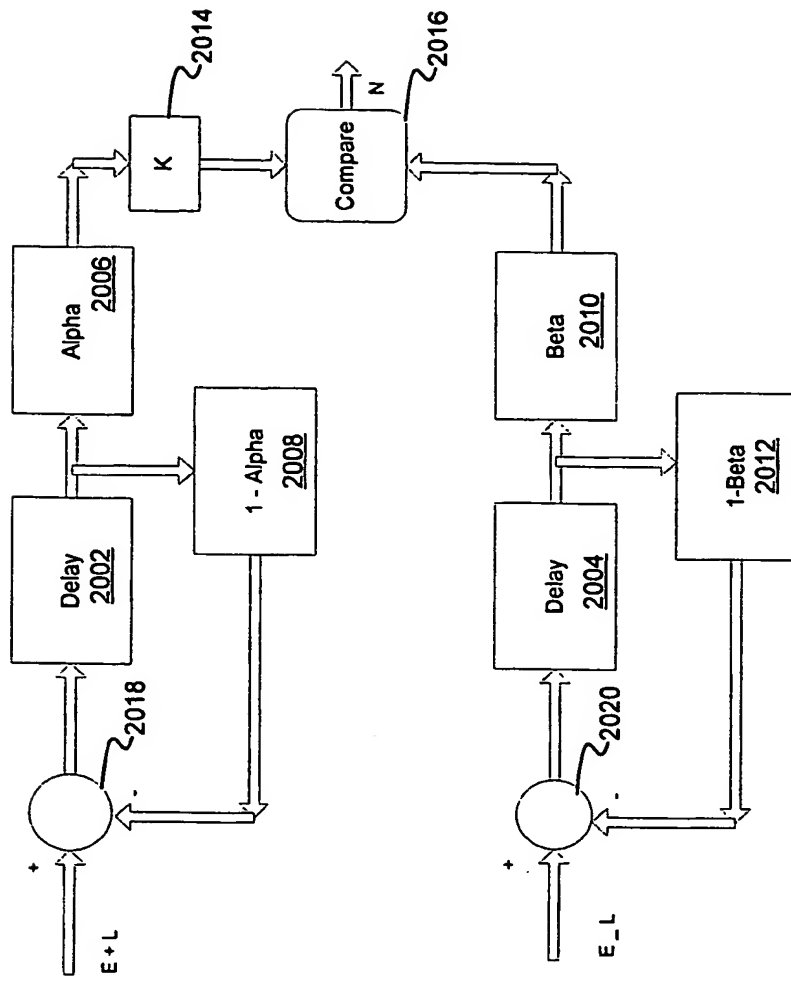


Figure 20

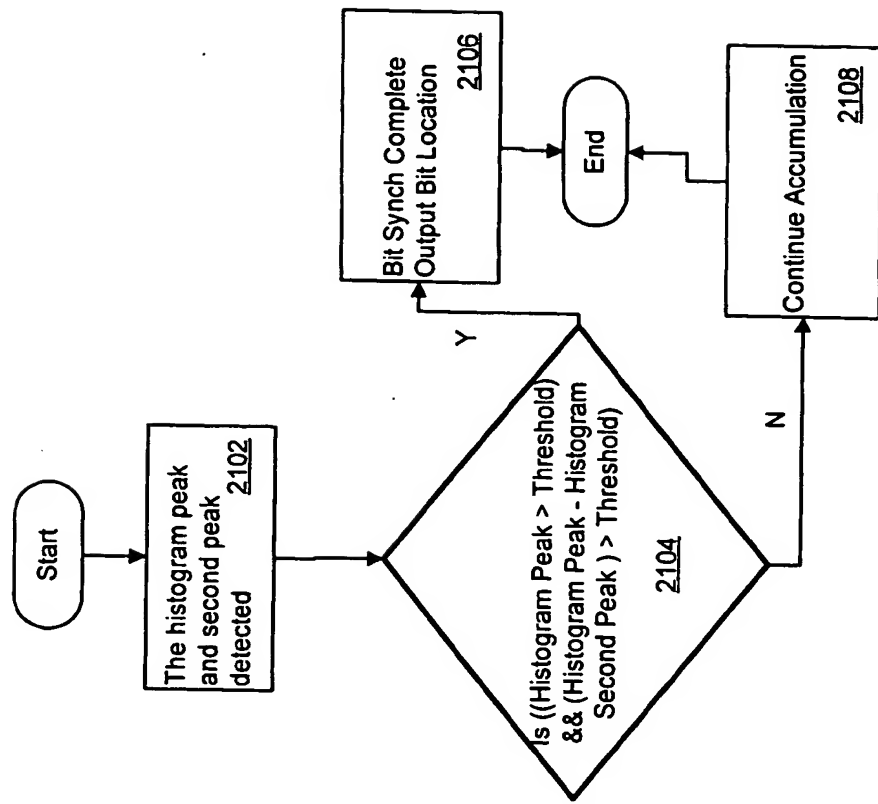
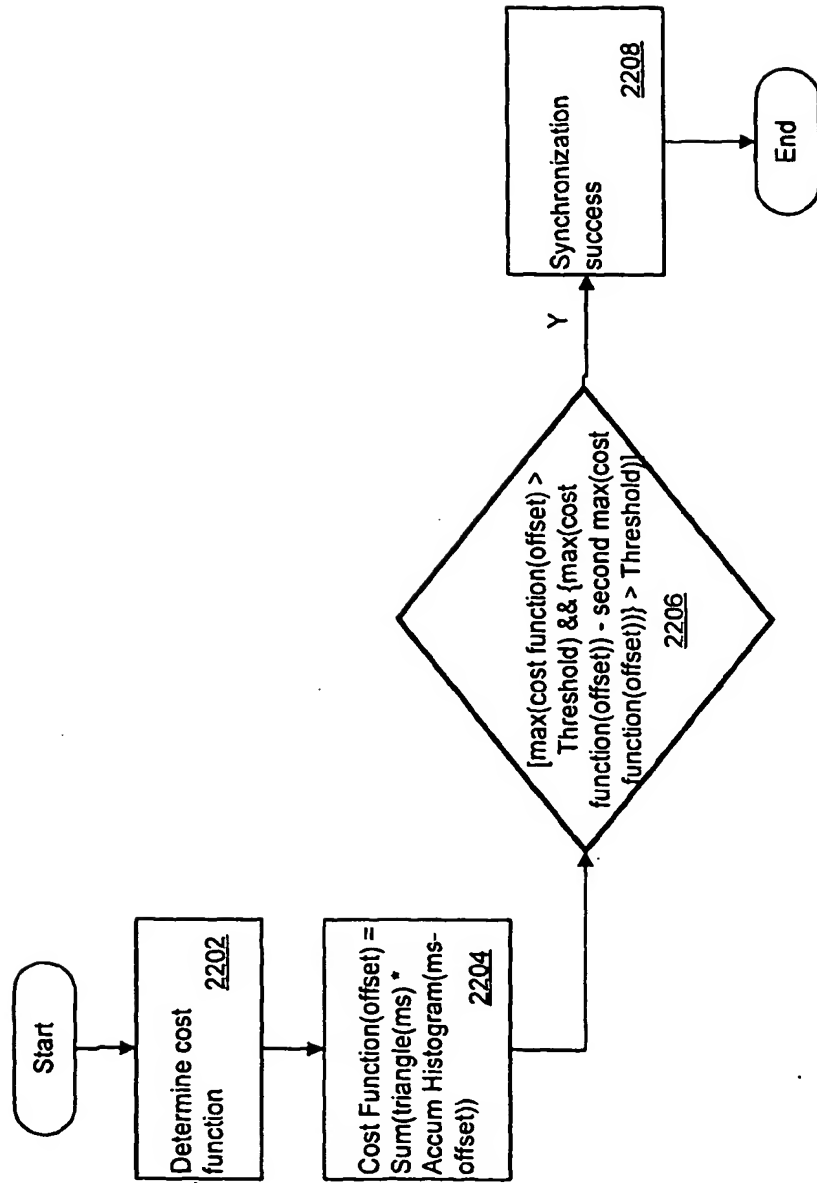


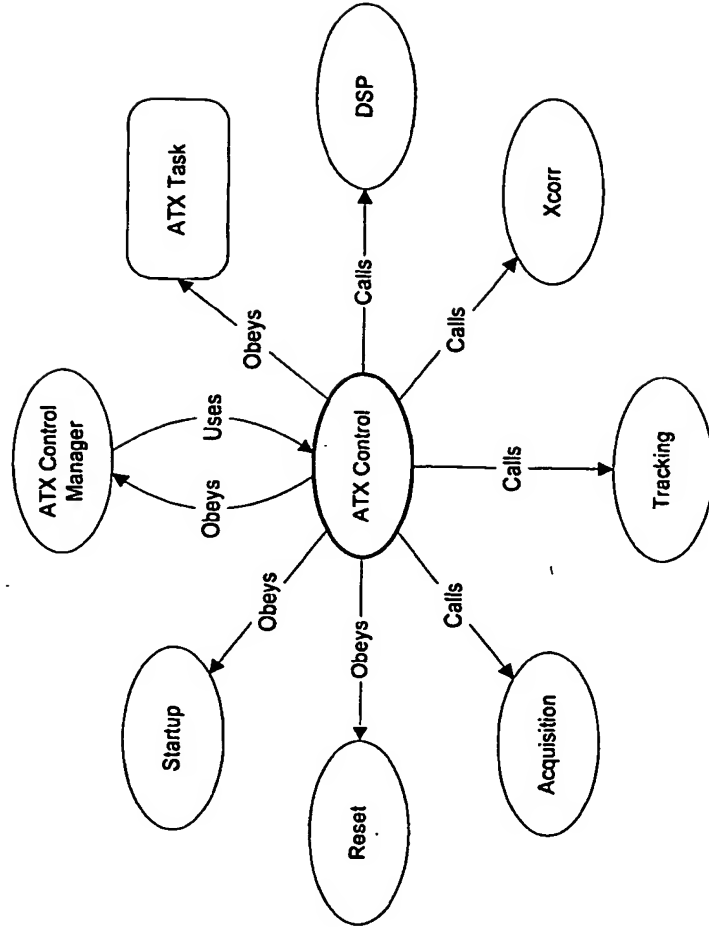
Figure 21

2100 ↗



2200 ↗

Figure 22



2300 ↗

Figure 23

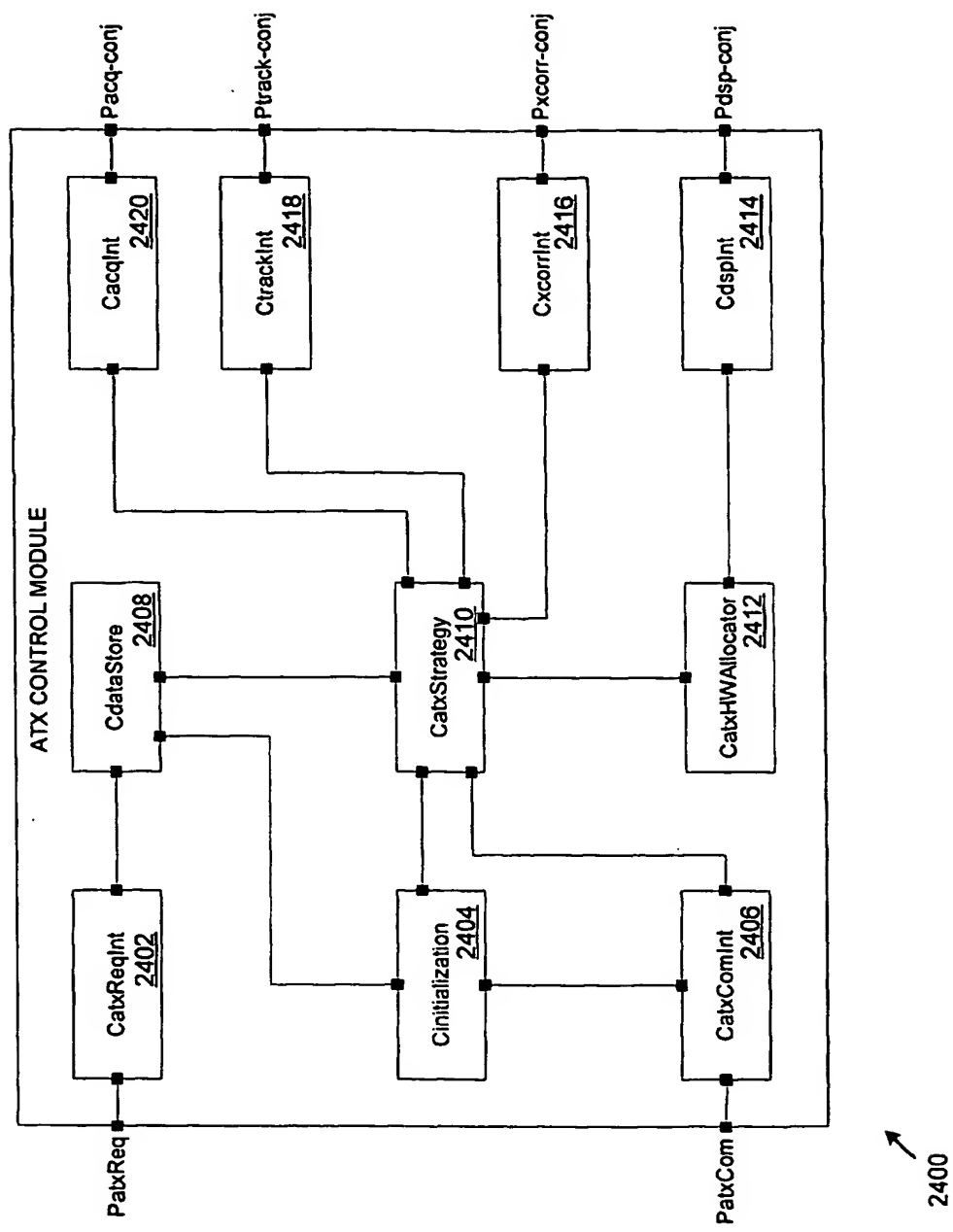


Figure 24

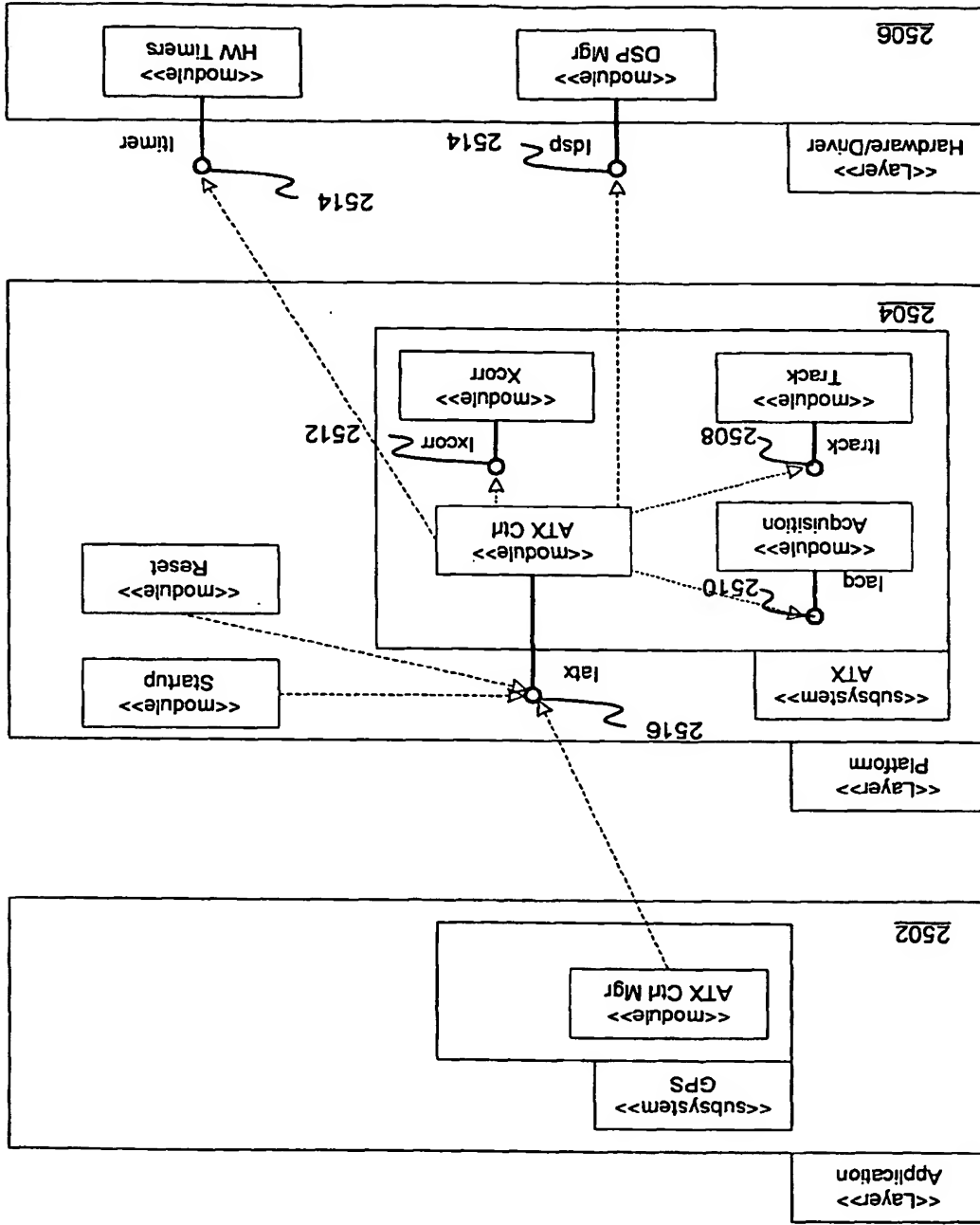
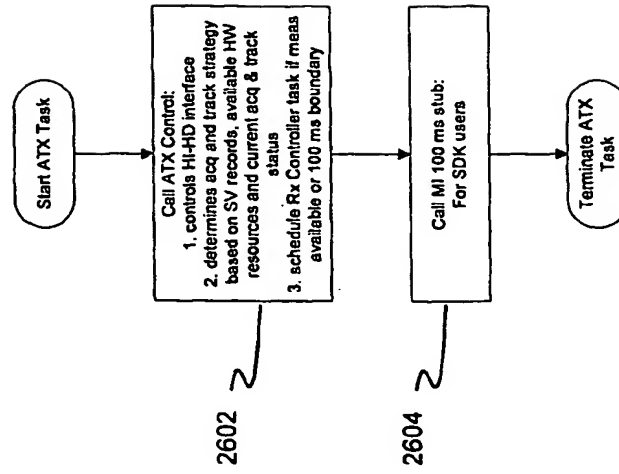


Figure 25



2600 ↗

Figure 26

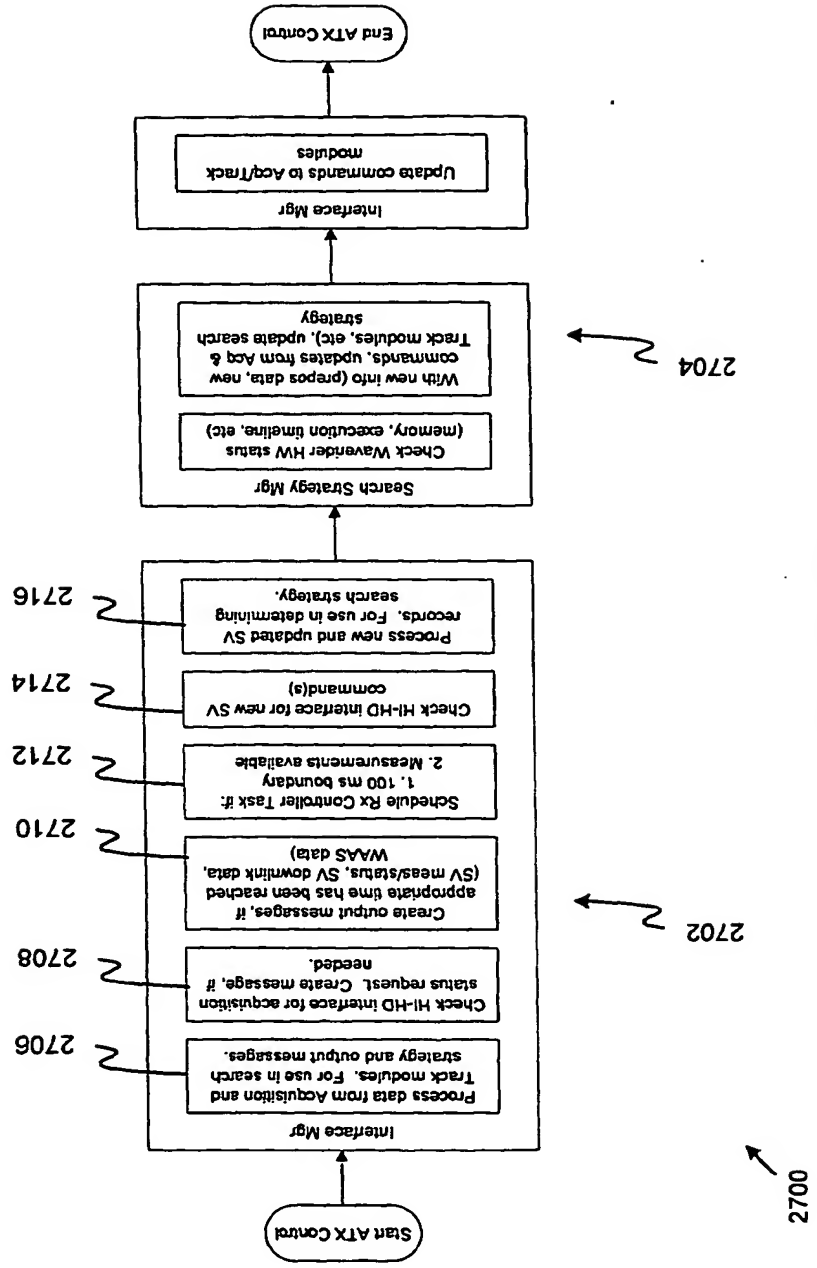
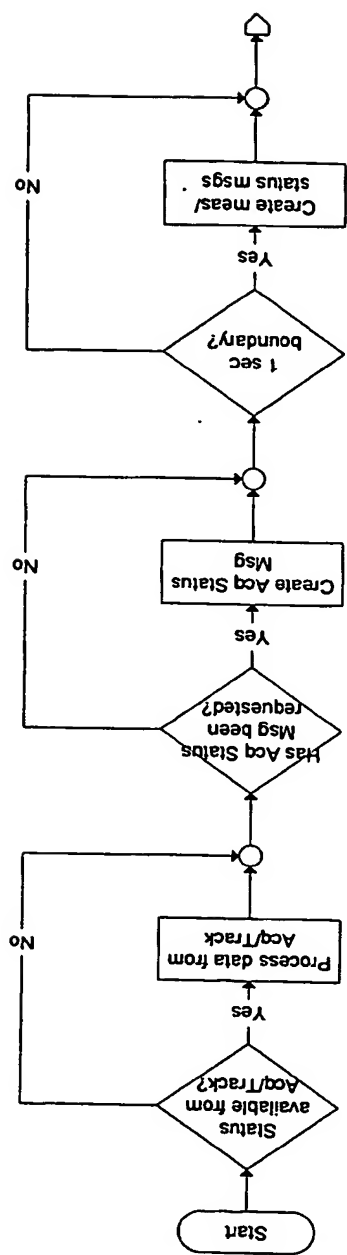


Figure 27



2800 ↗

Figure 28

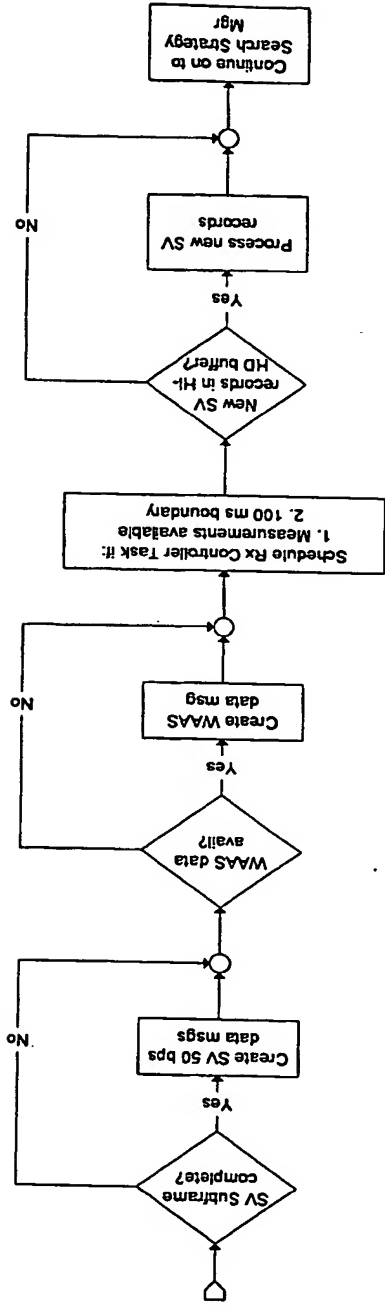


Figure 29

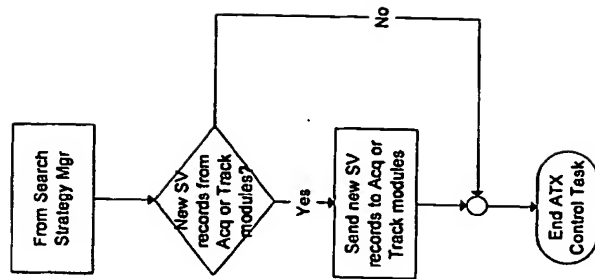
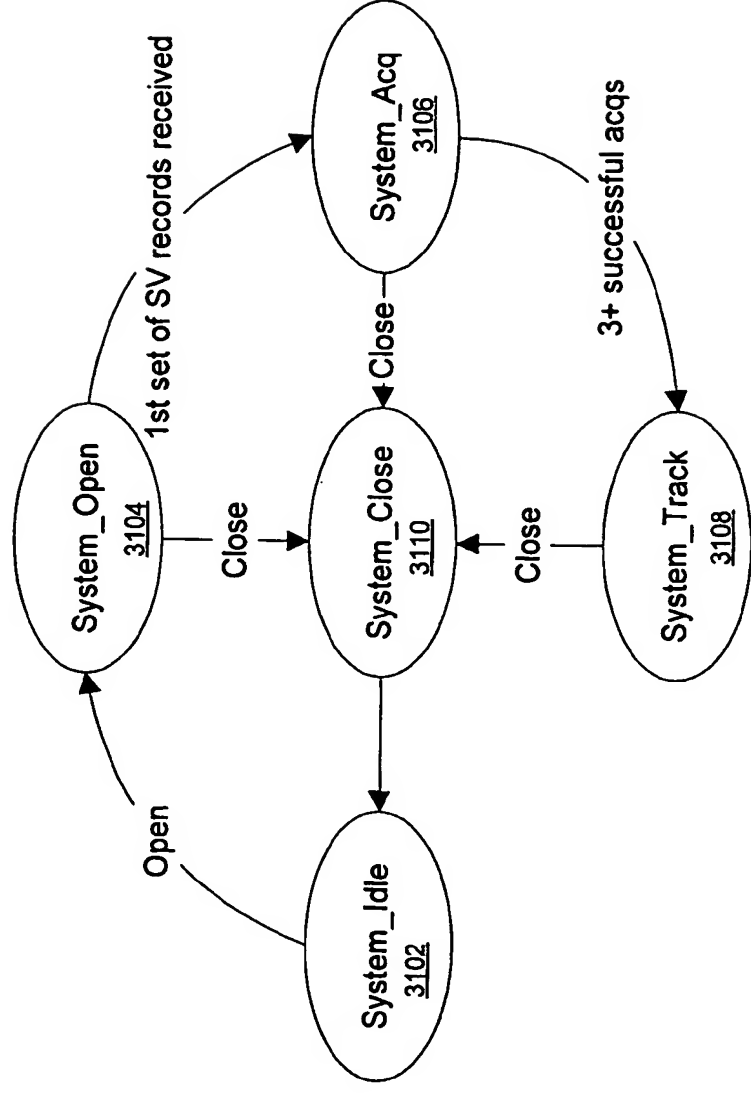


Figure 30



3100 ↗

Figure 31

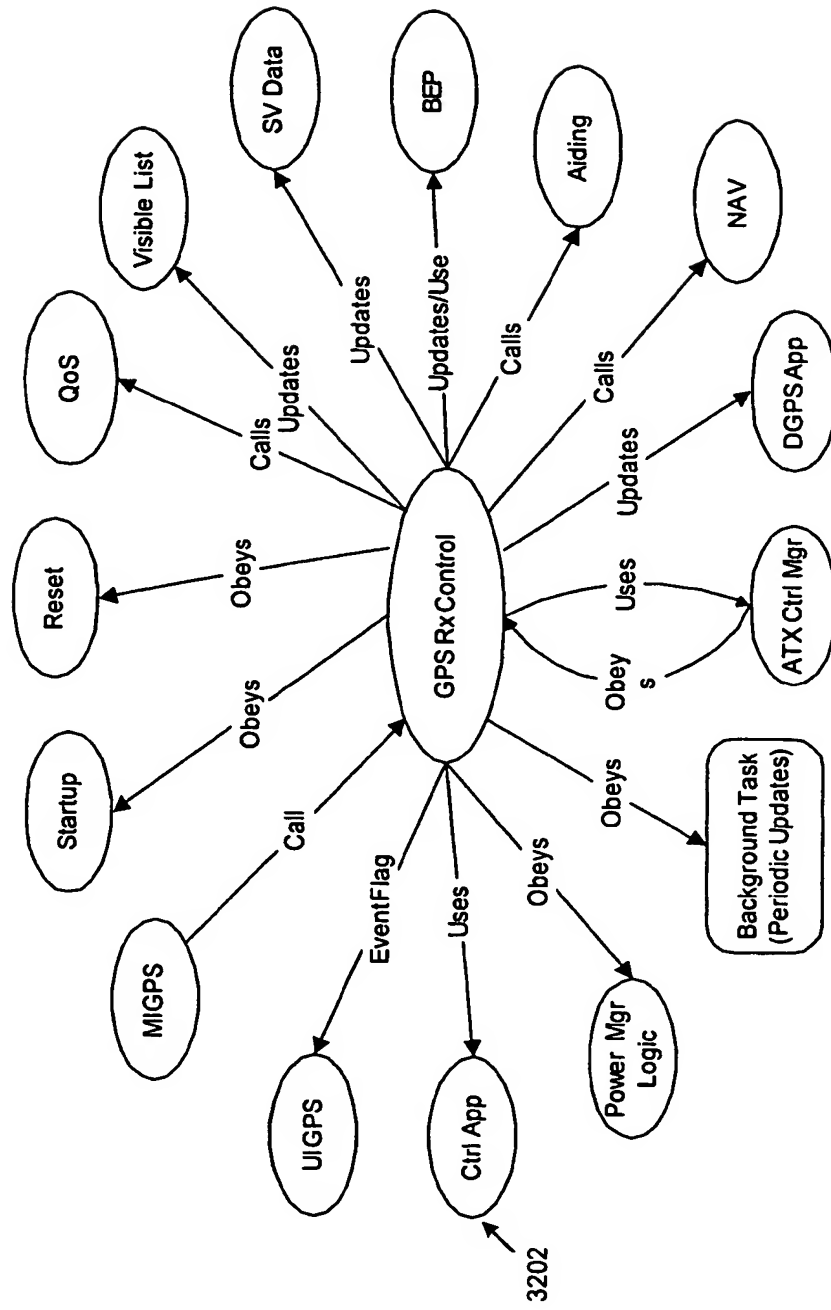


Figure 32

3200

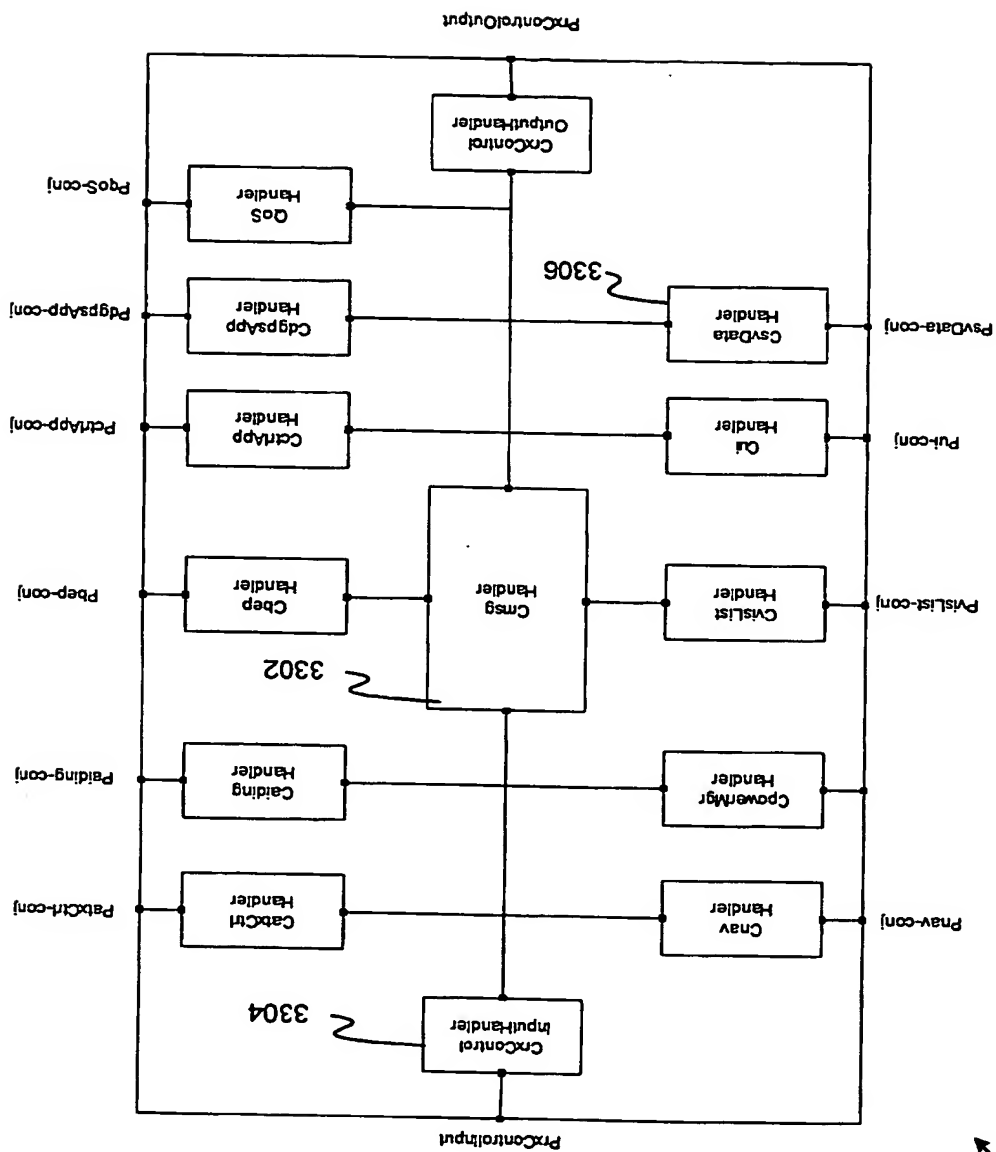
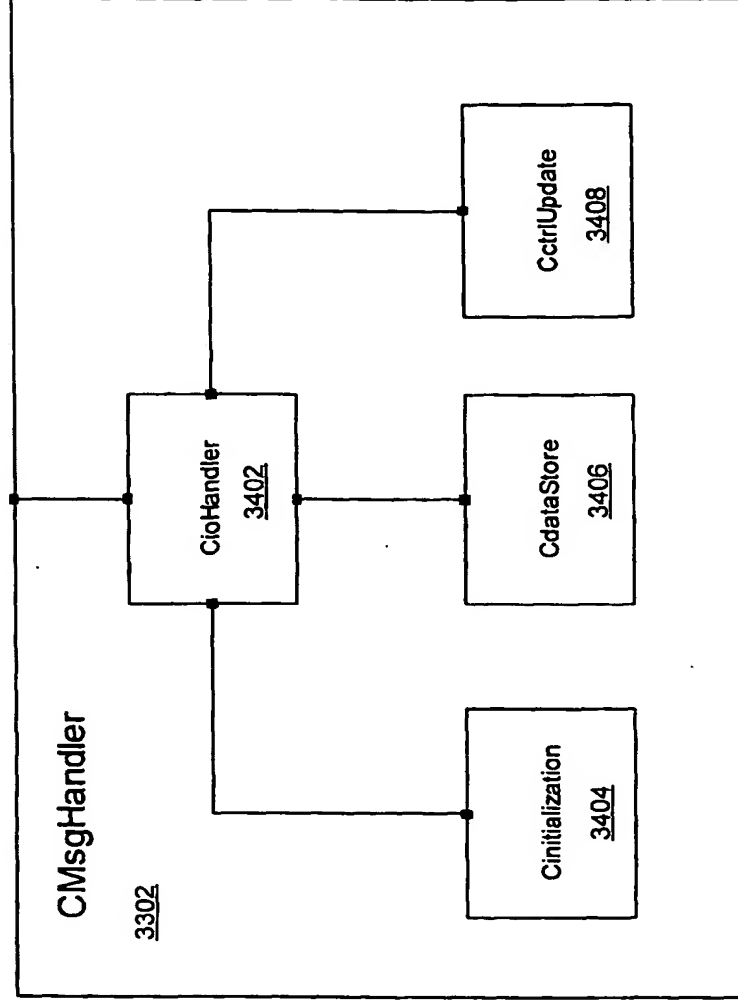


Figure 33



3400 ↗

Figure 34

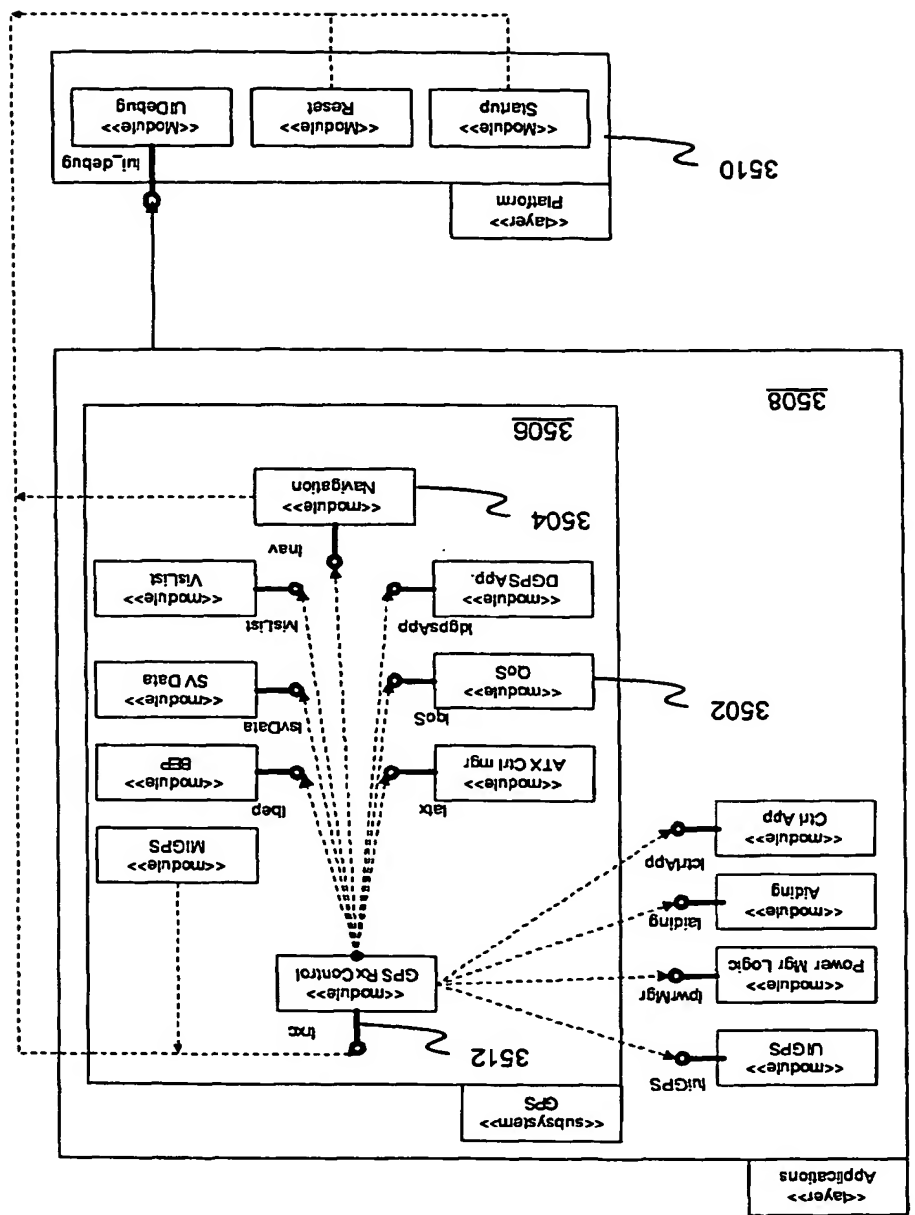
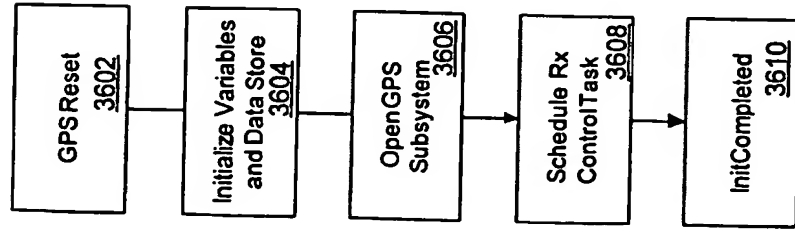
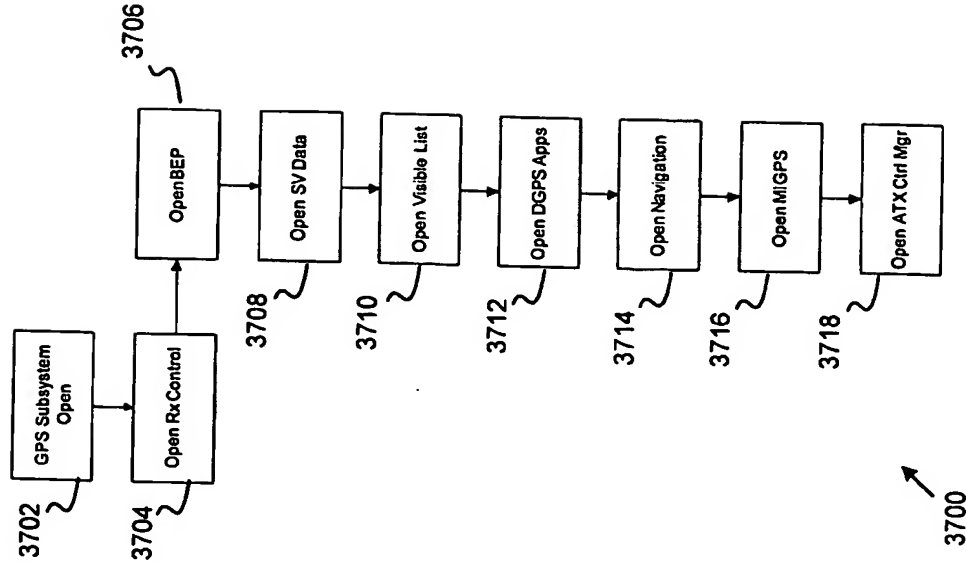


Figure 35



3600

Figure 36



3700

Figure 37

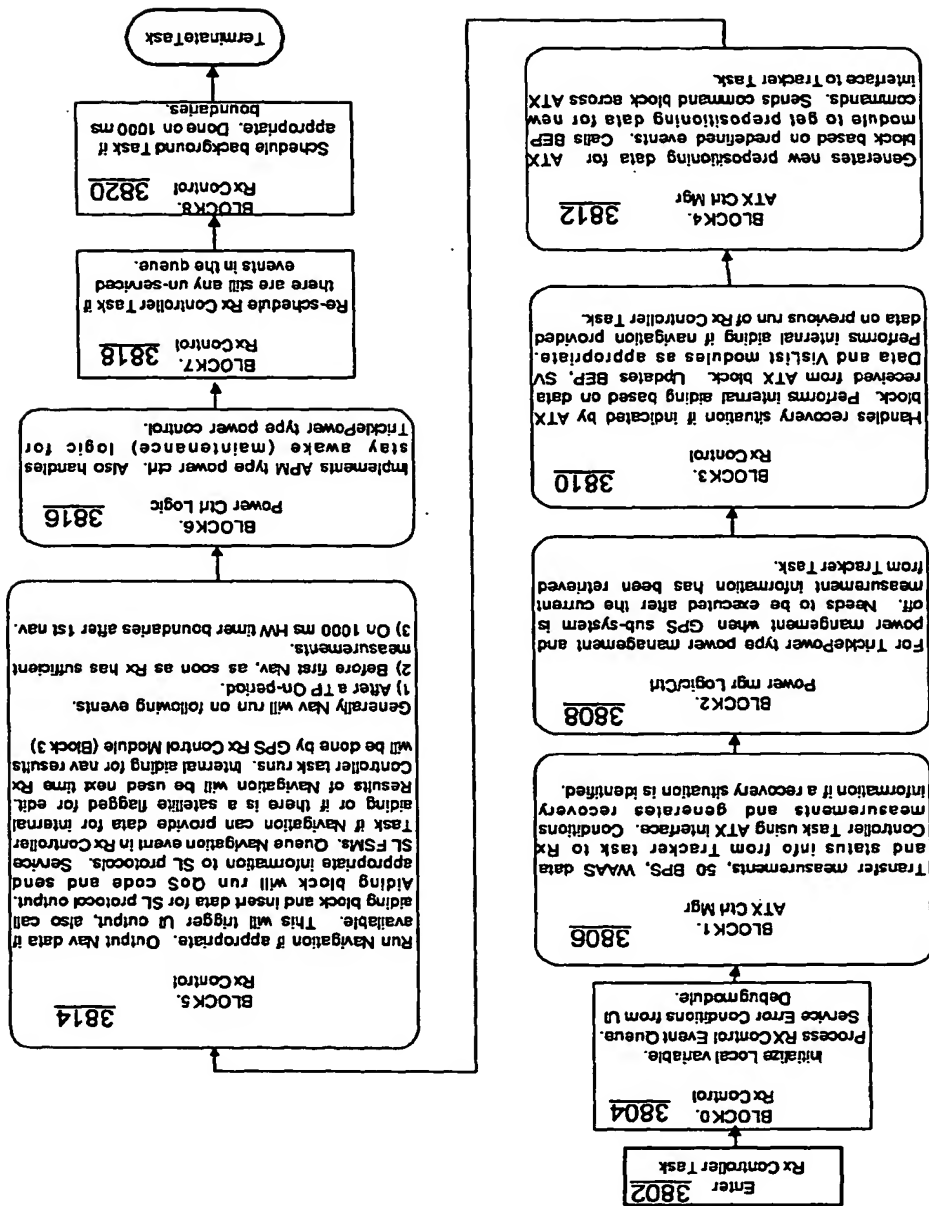


Figure 38

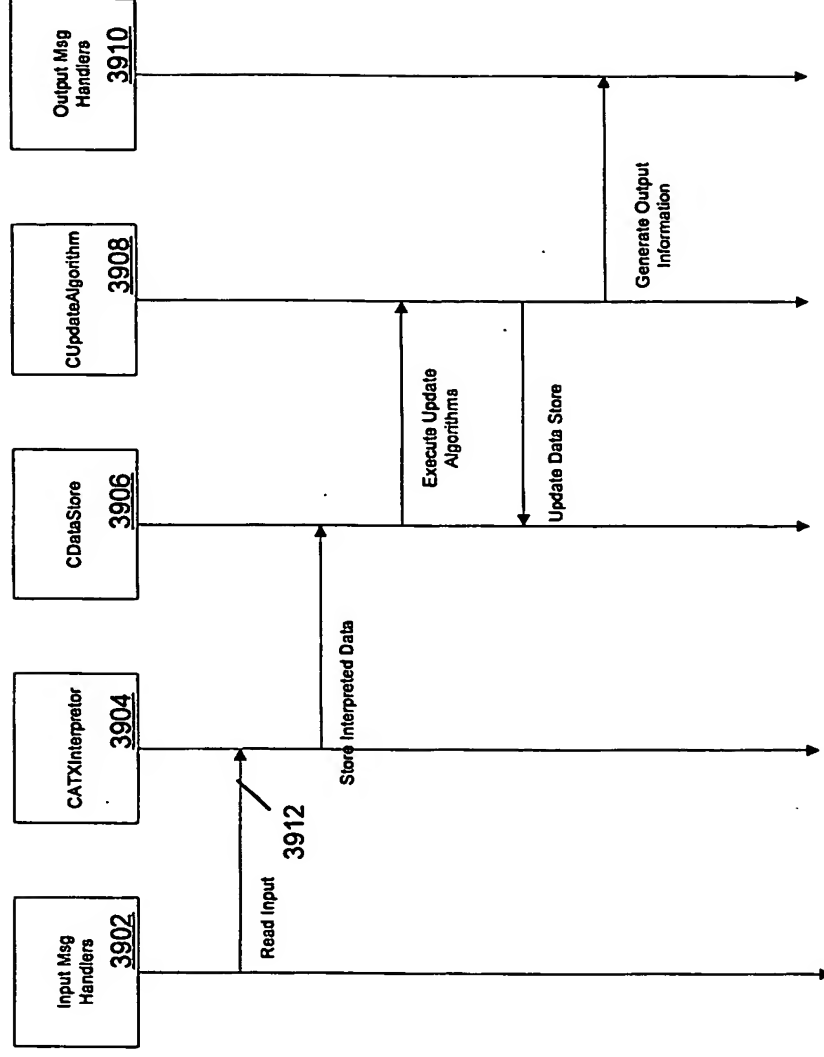


Figure 39

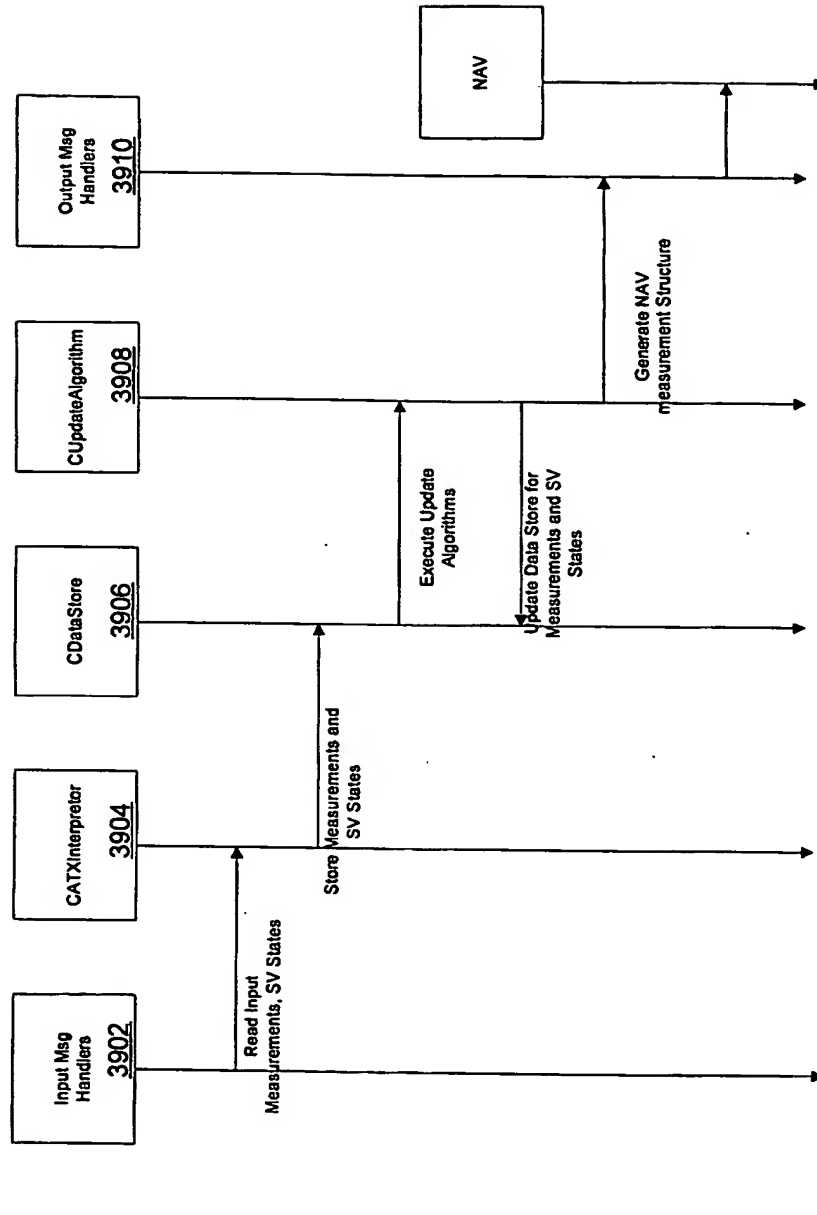


Figure 40

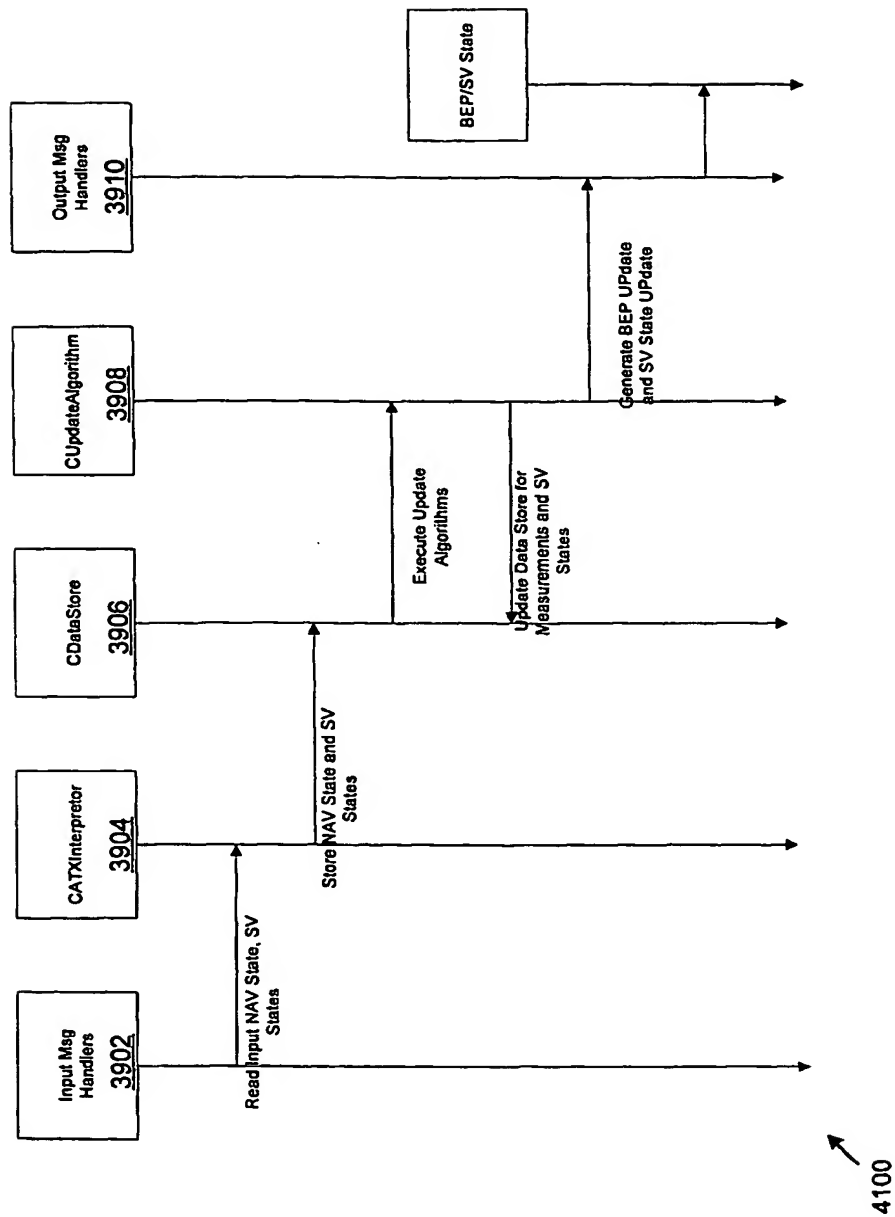


Figure 41

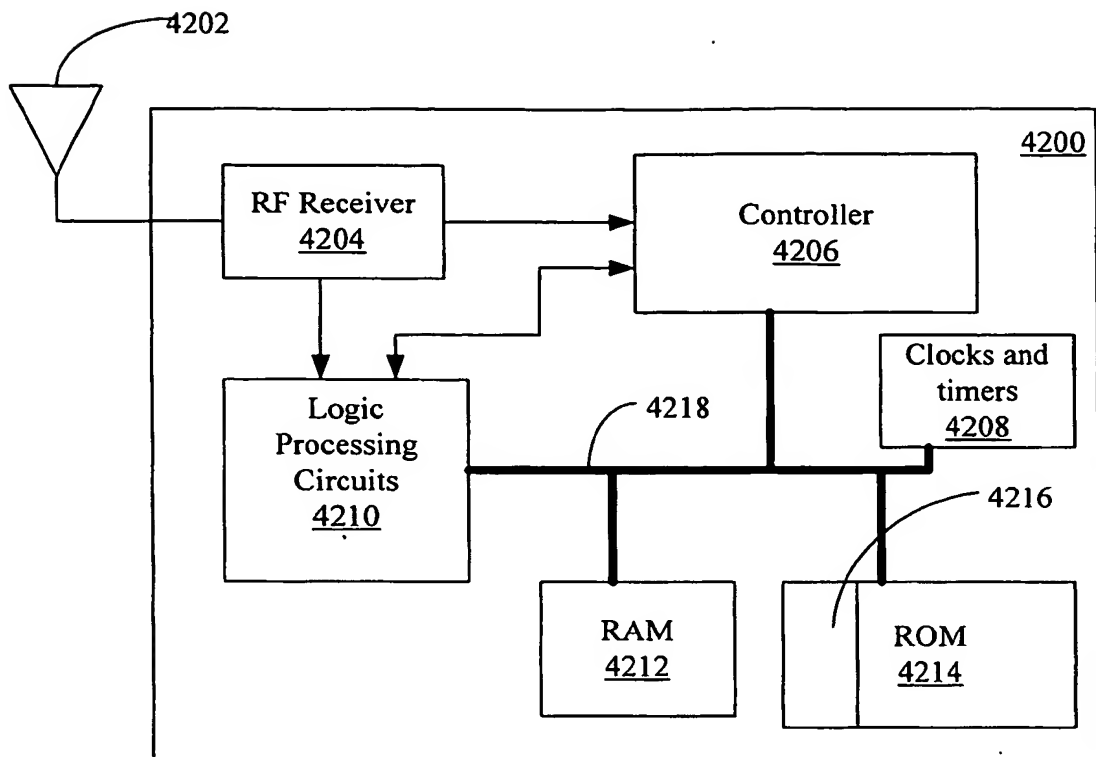


FIG. 42

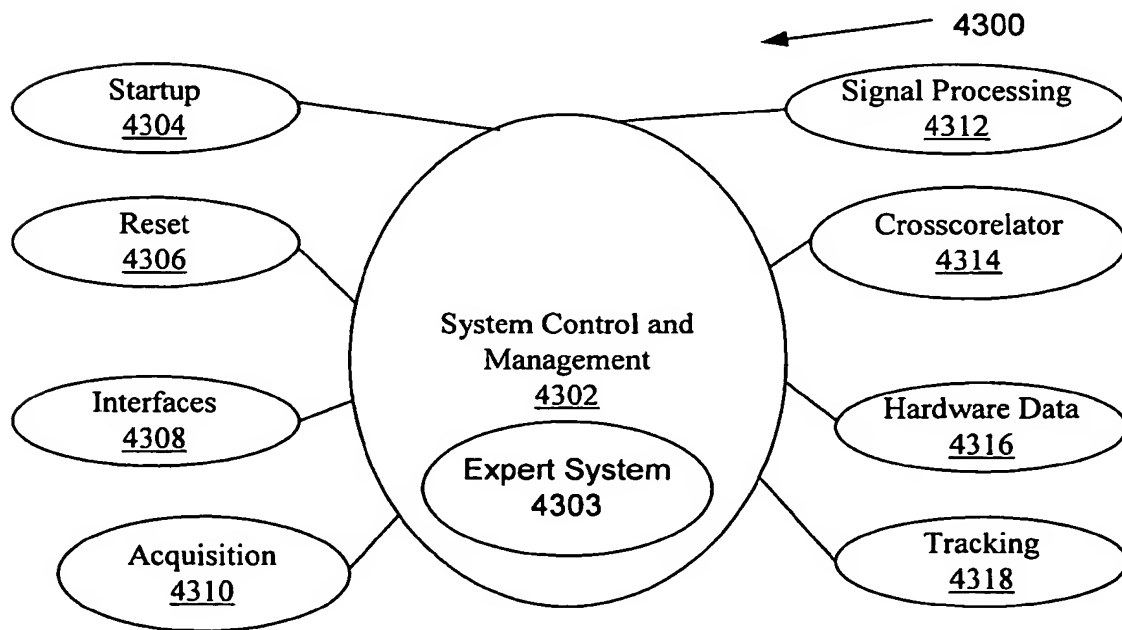


FIG. 43

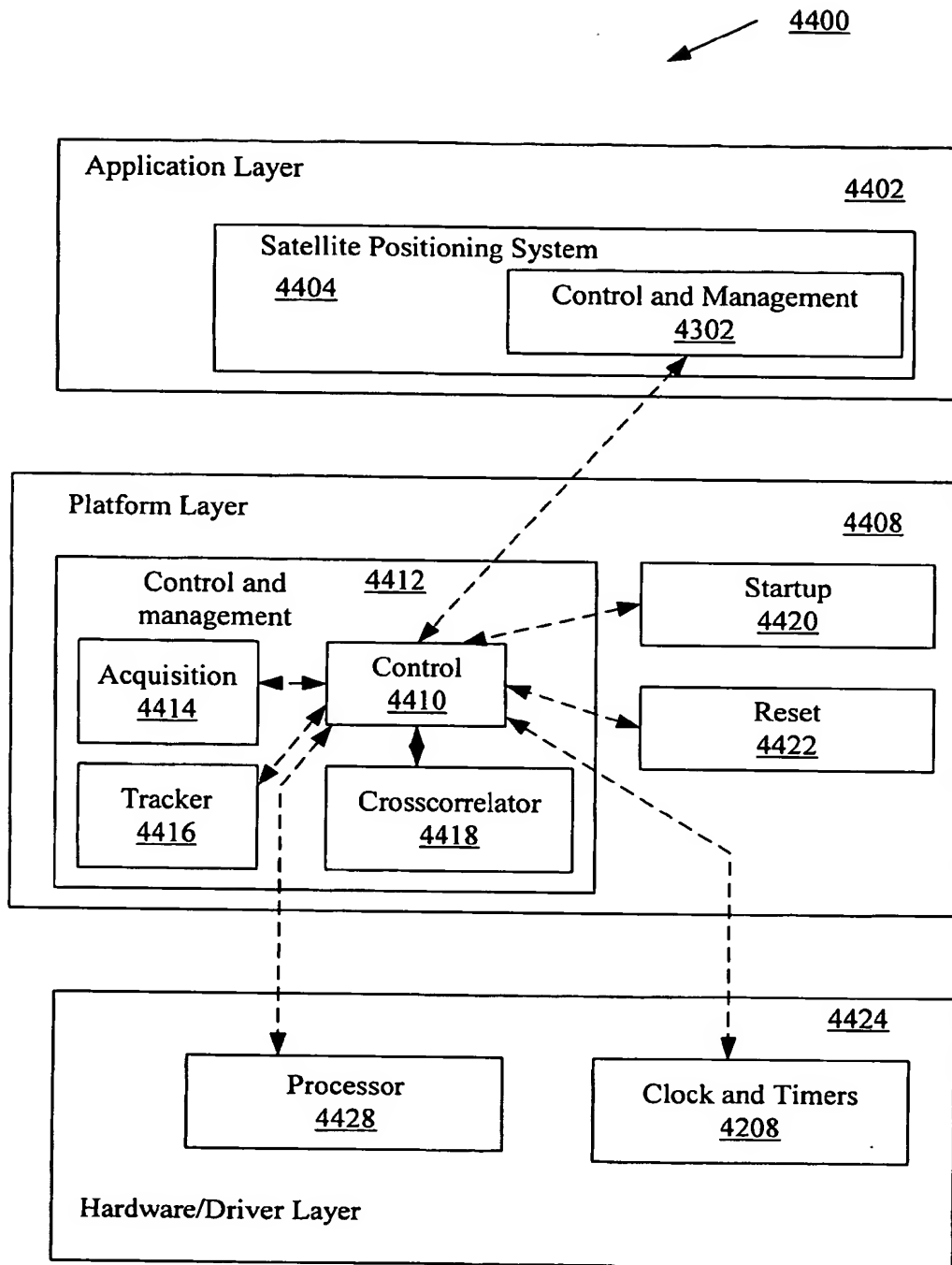


FIG. 44

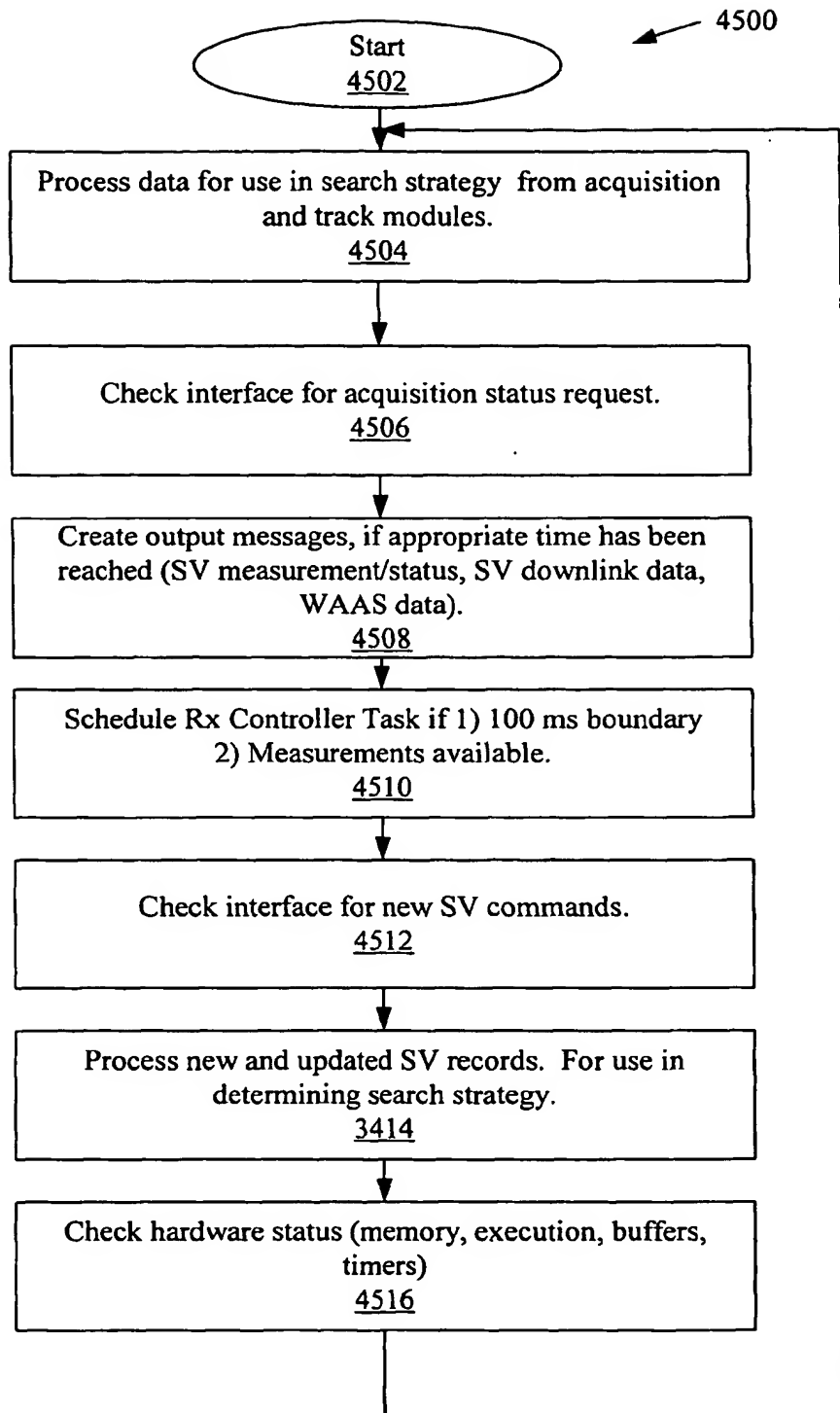


FIG.45

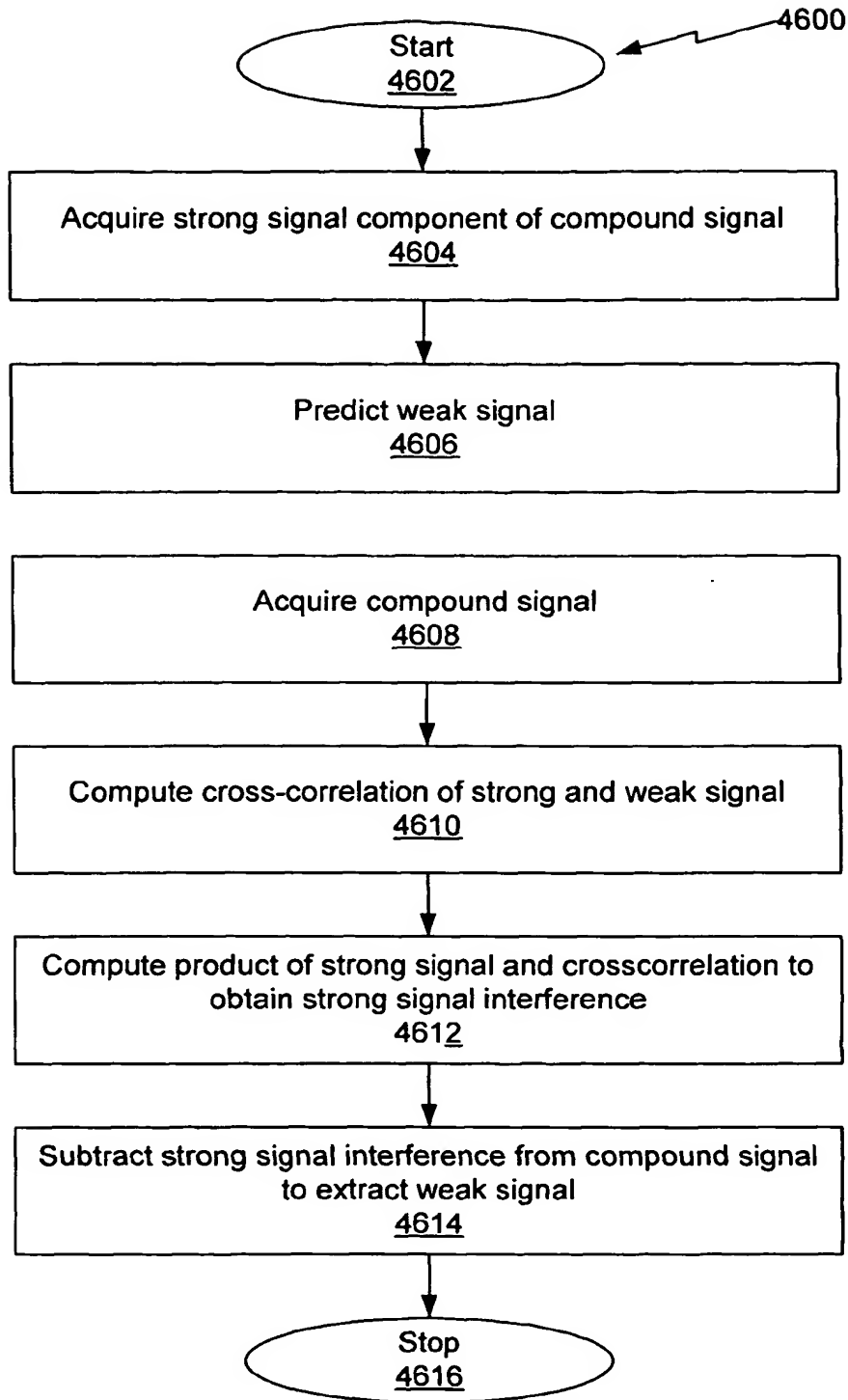


FIG. 46

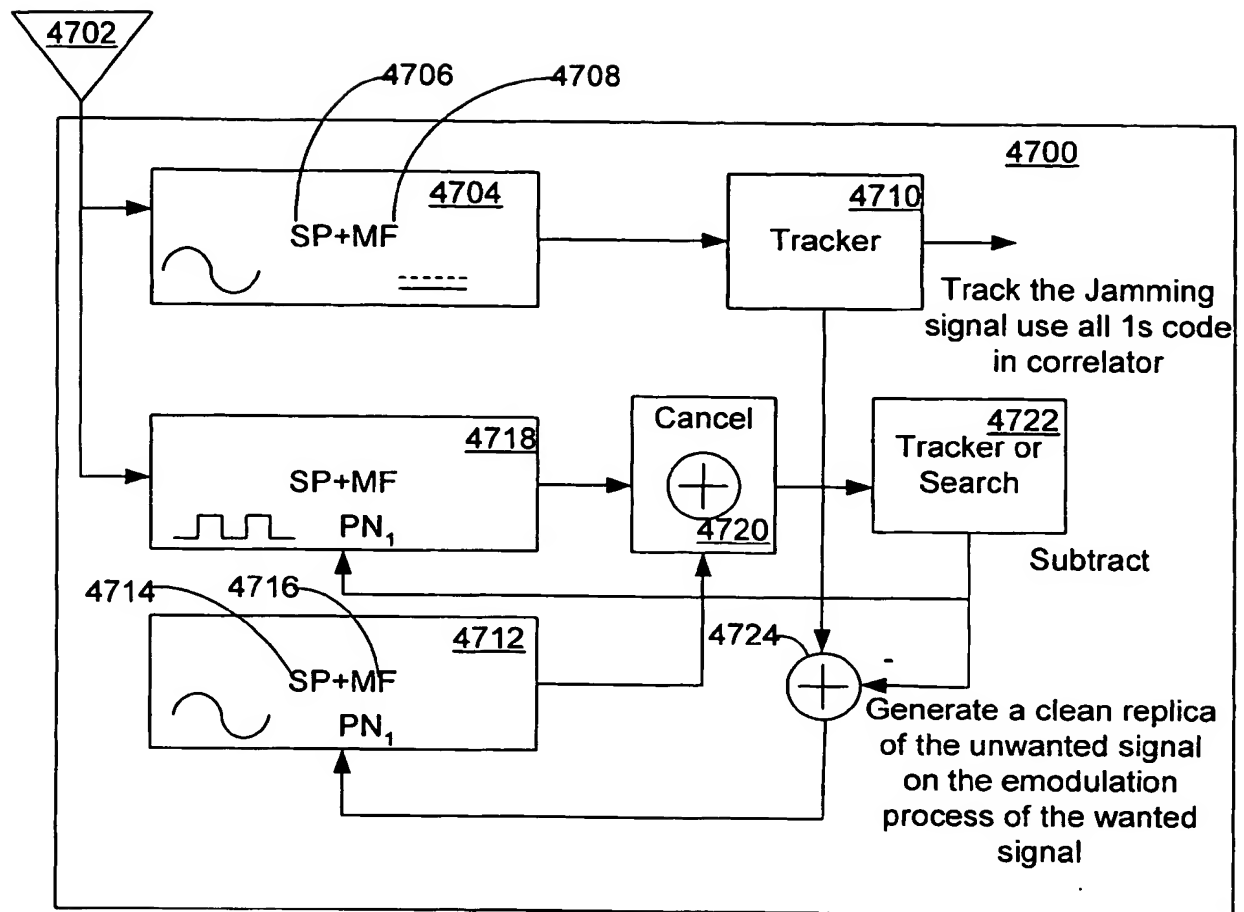


FIG. 47

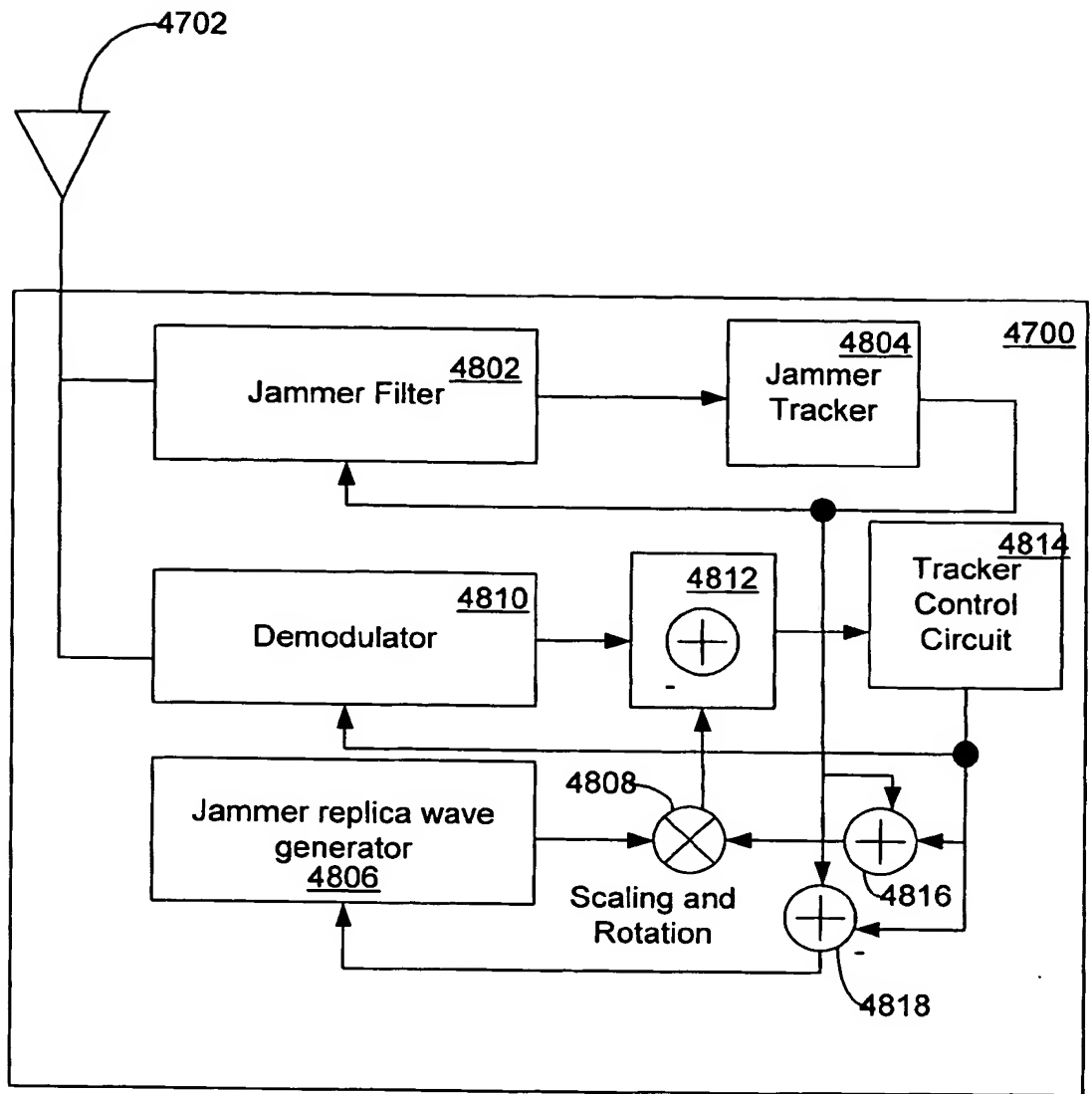


FIG. 48

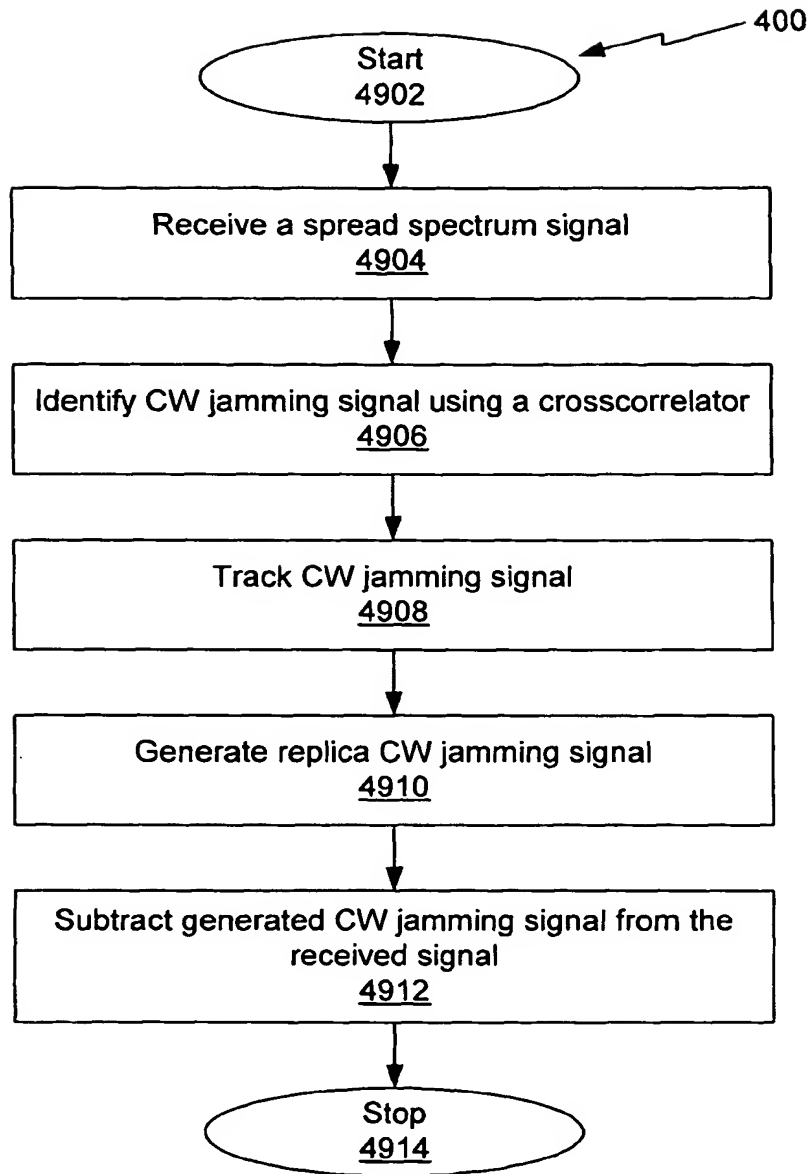


FIG. 49

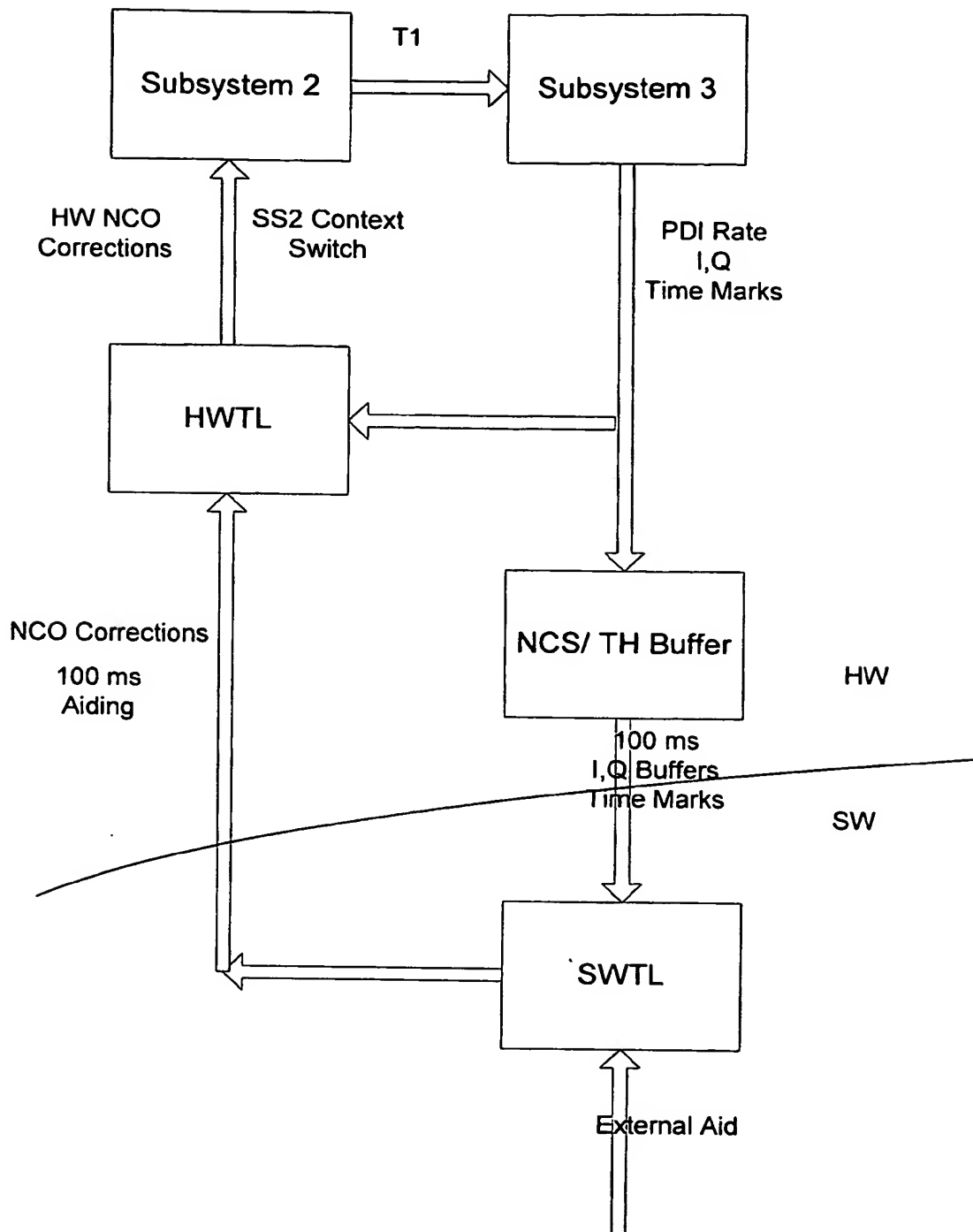


Figure 50

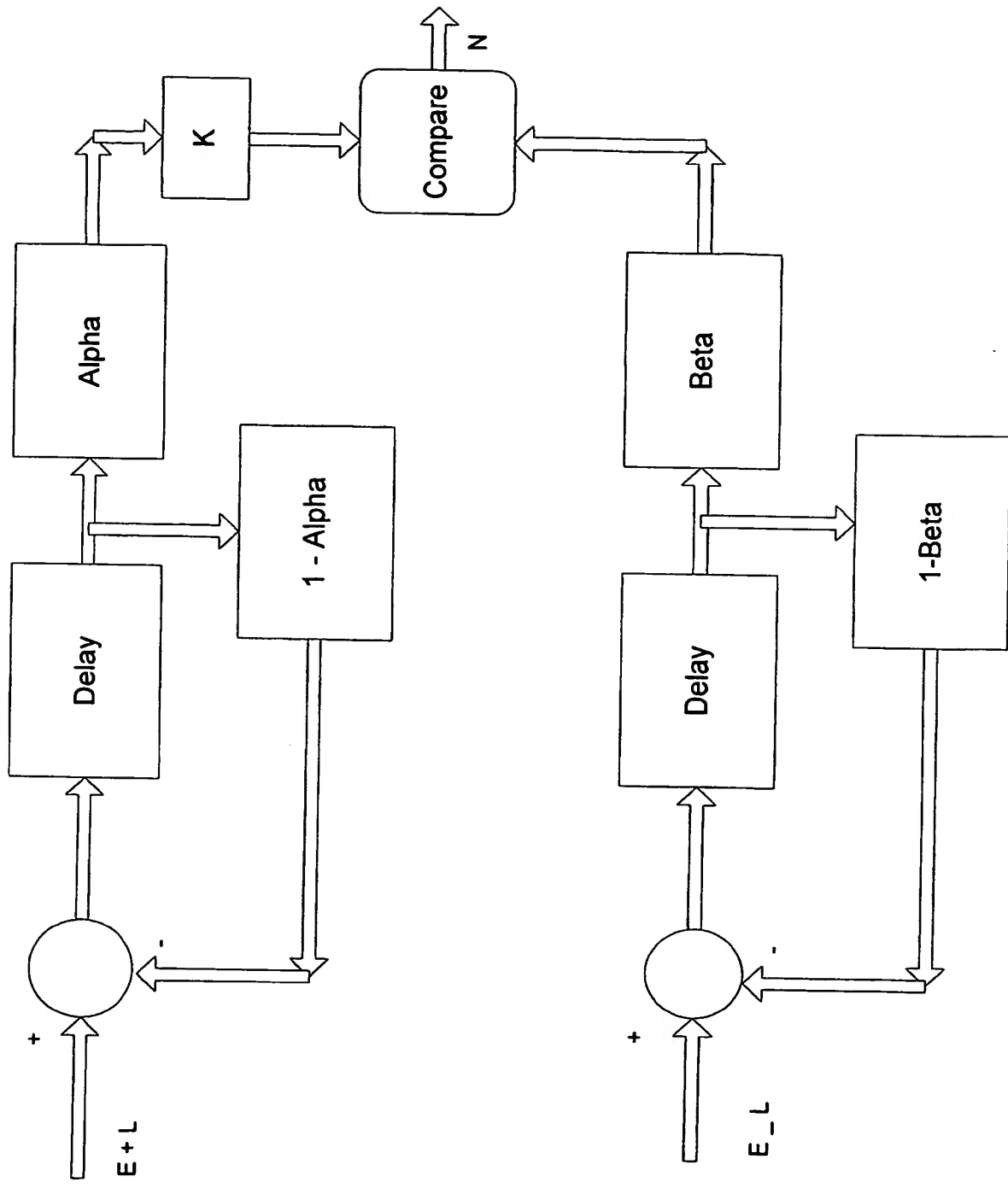


Figure 51

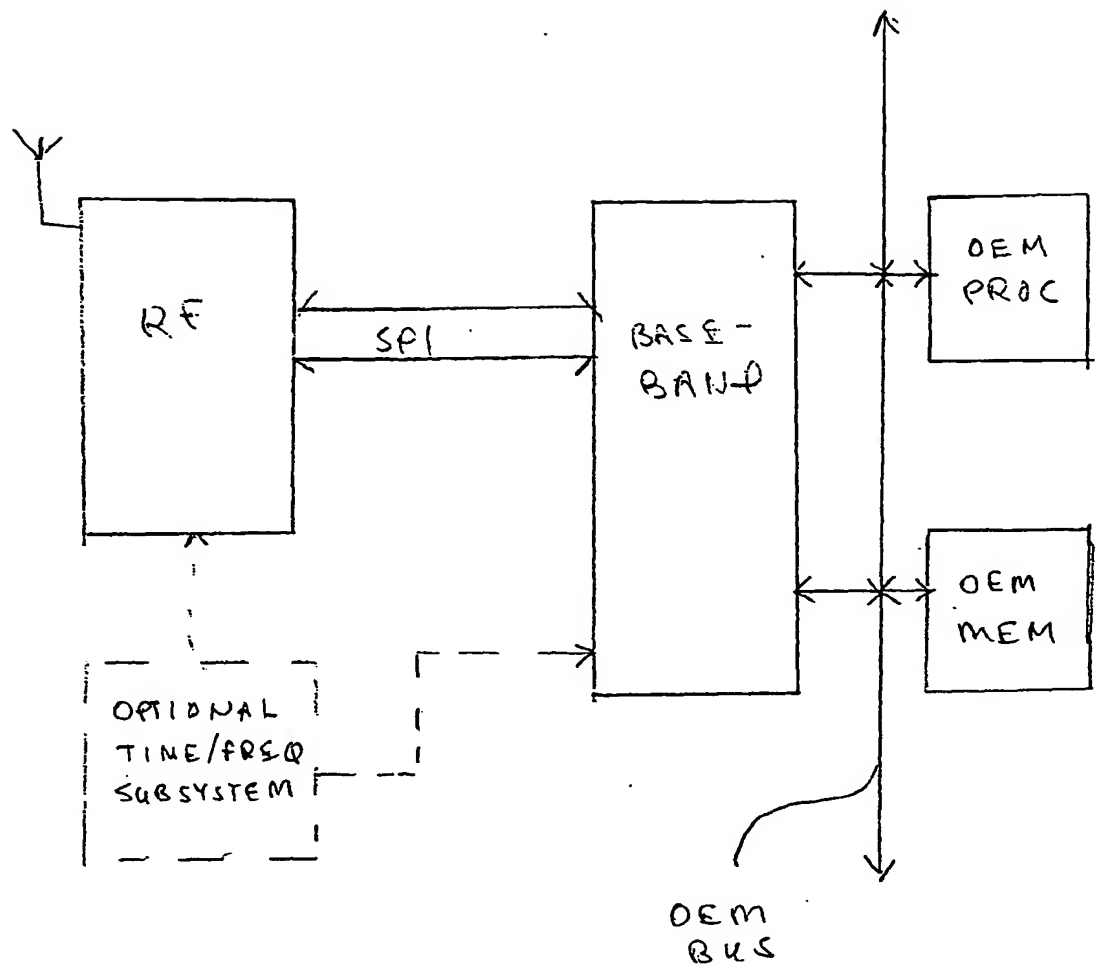


FIG. 1
PART II

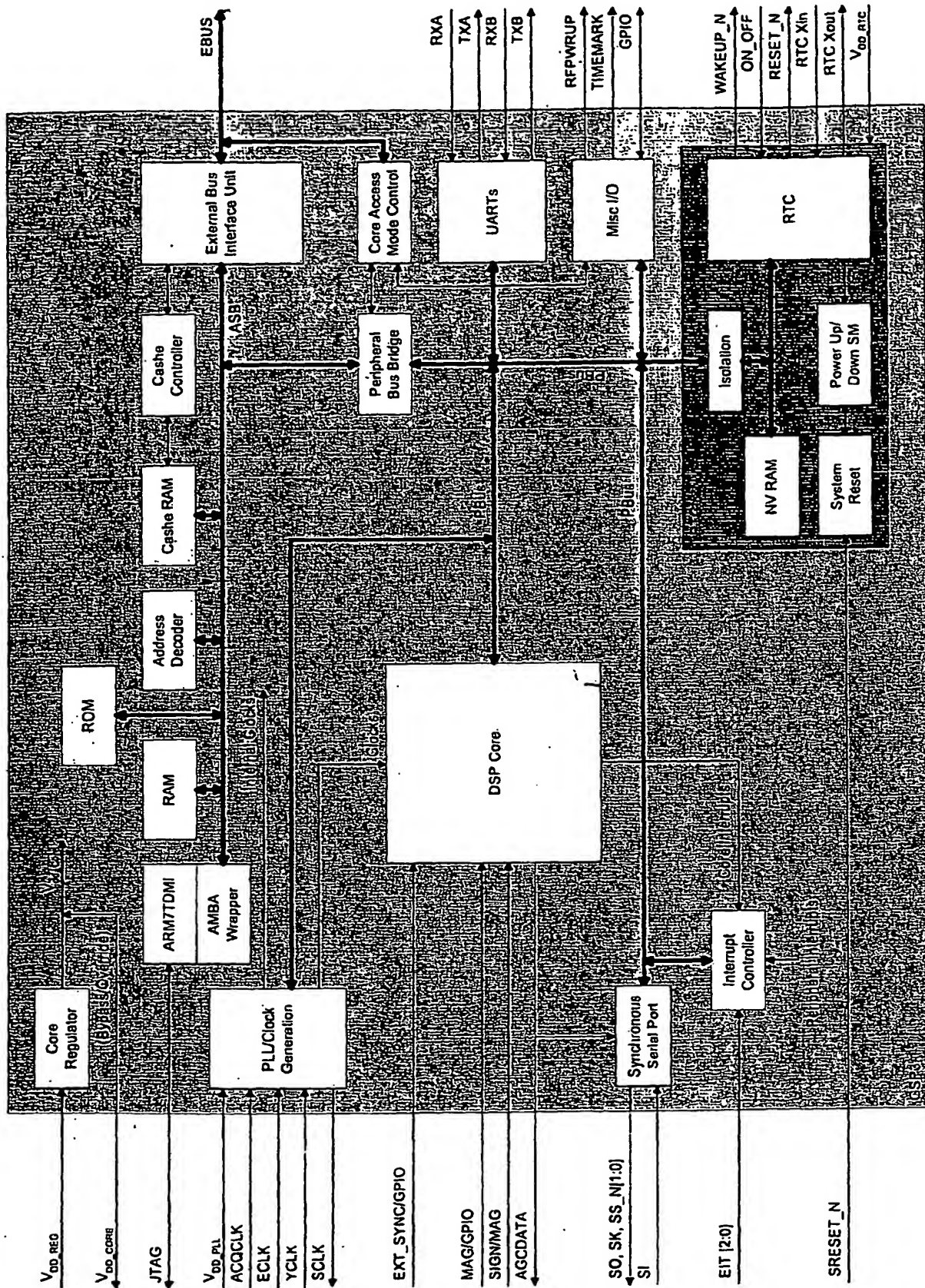
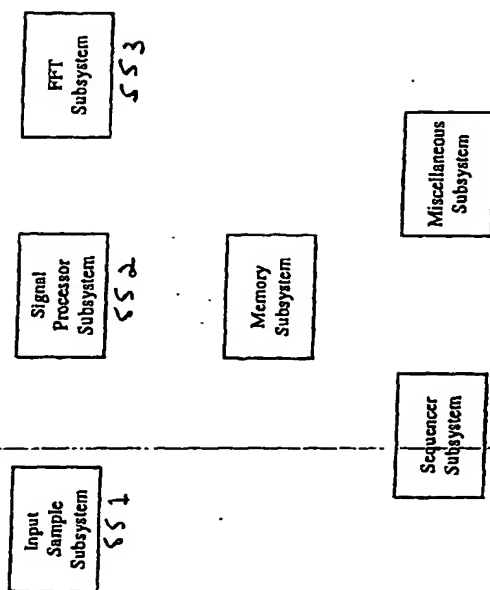


Fig. 1A
D0077



SiRFStar3 Core Major Subsystem

F16.2
PART II

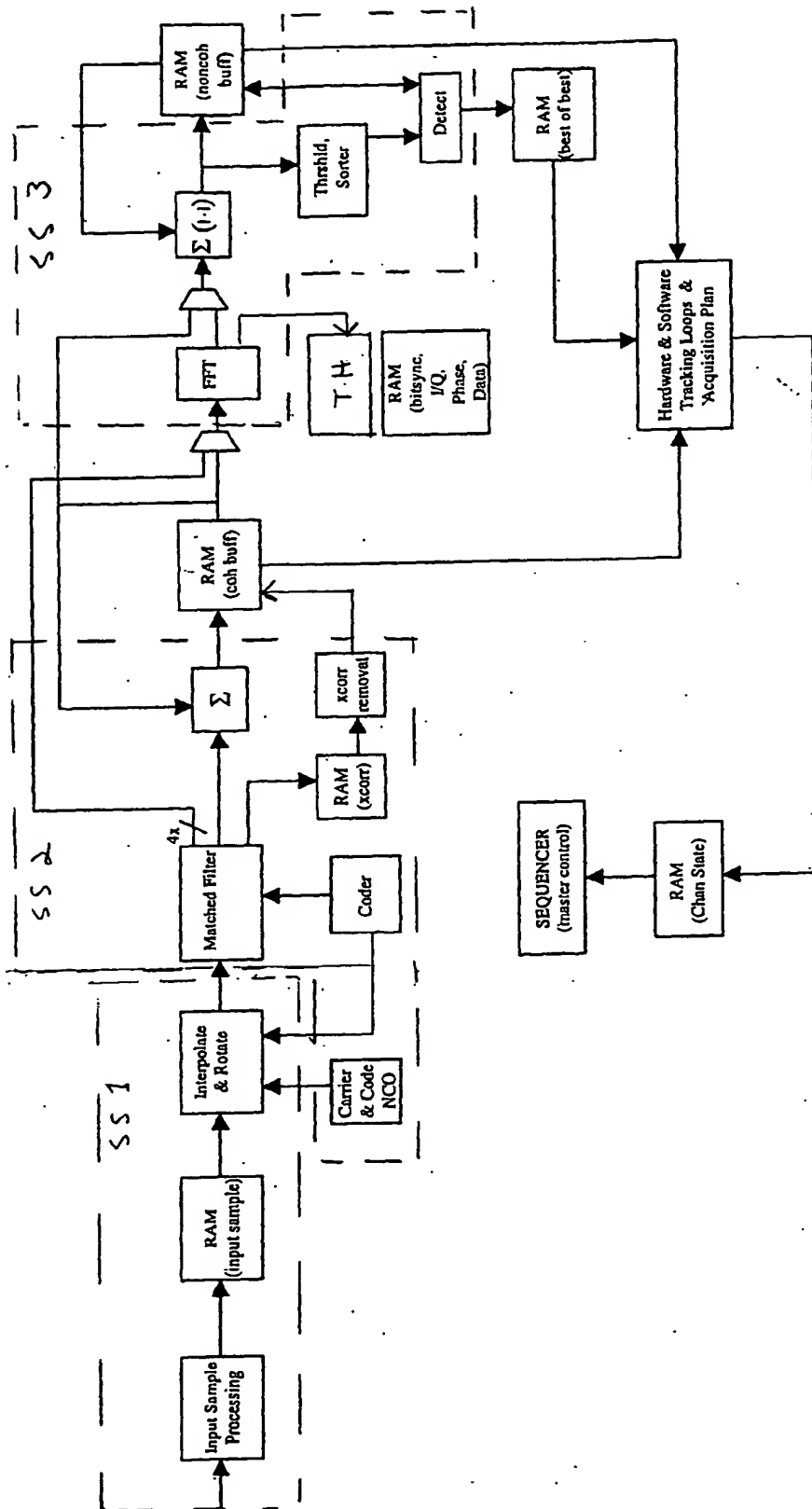
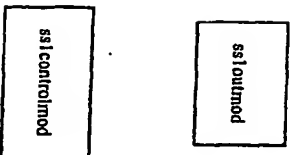
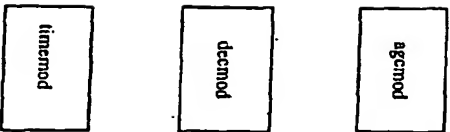


Fig. 4
PART II

General Data Flow



F-16.5

PART II

input sample subsystem partitioning

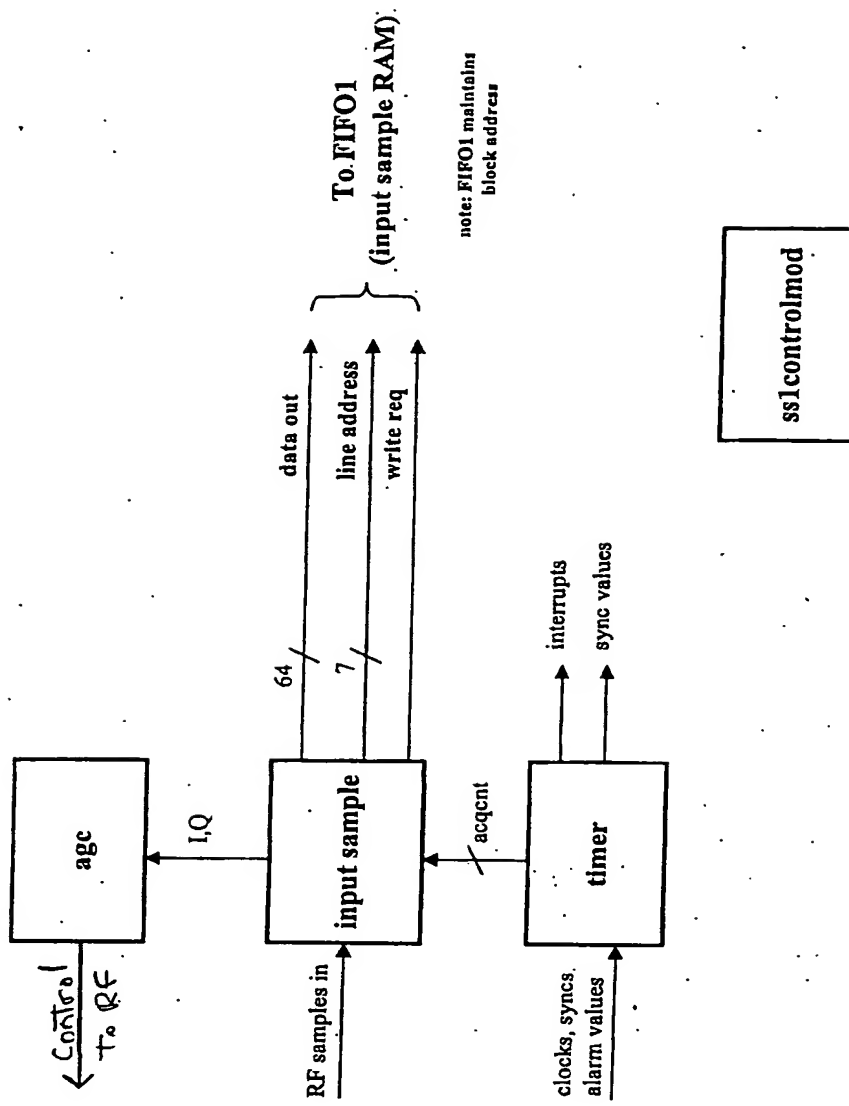
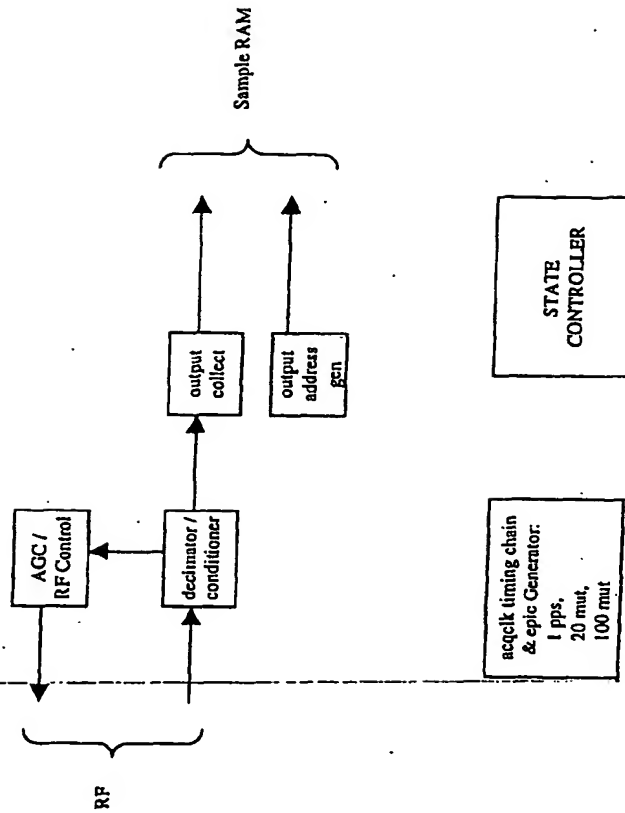


Fig. 4
PART II

Subsystem 1 Partitioning



acqclk timing chain
& epic Generator:
1 pps,
20 mut,
100 mut

STATE
CONTROLLER

Fig. 7
PART II

input sample subsystem flow

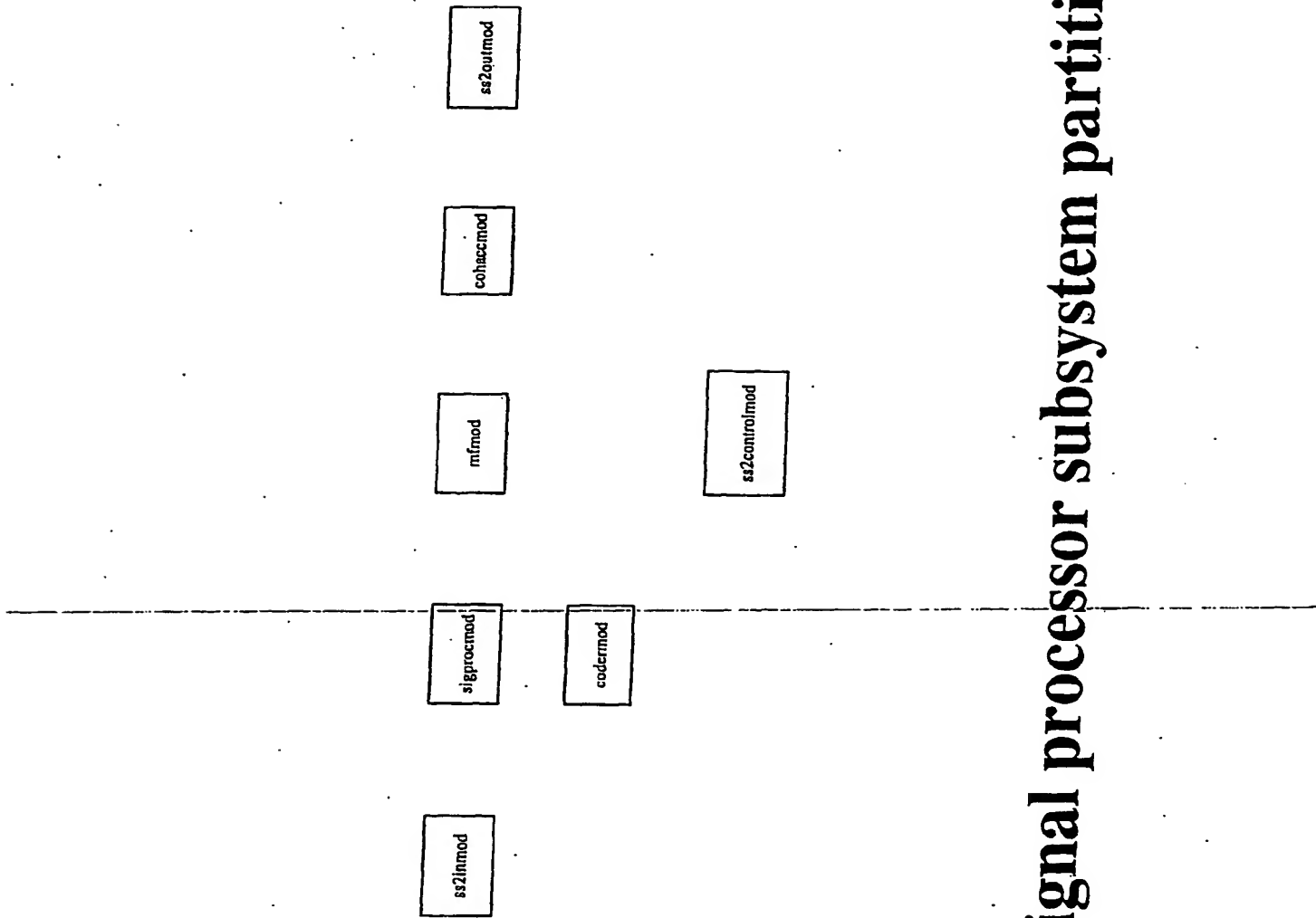


Fig. 8
PART II
signal processor subsystem partitioning

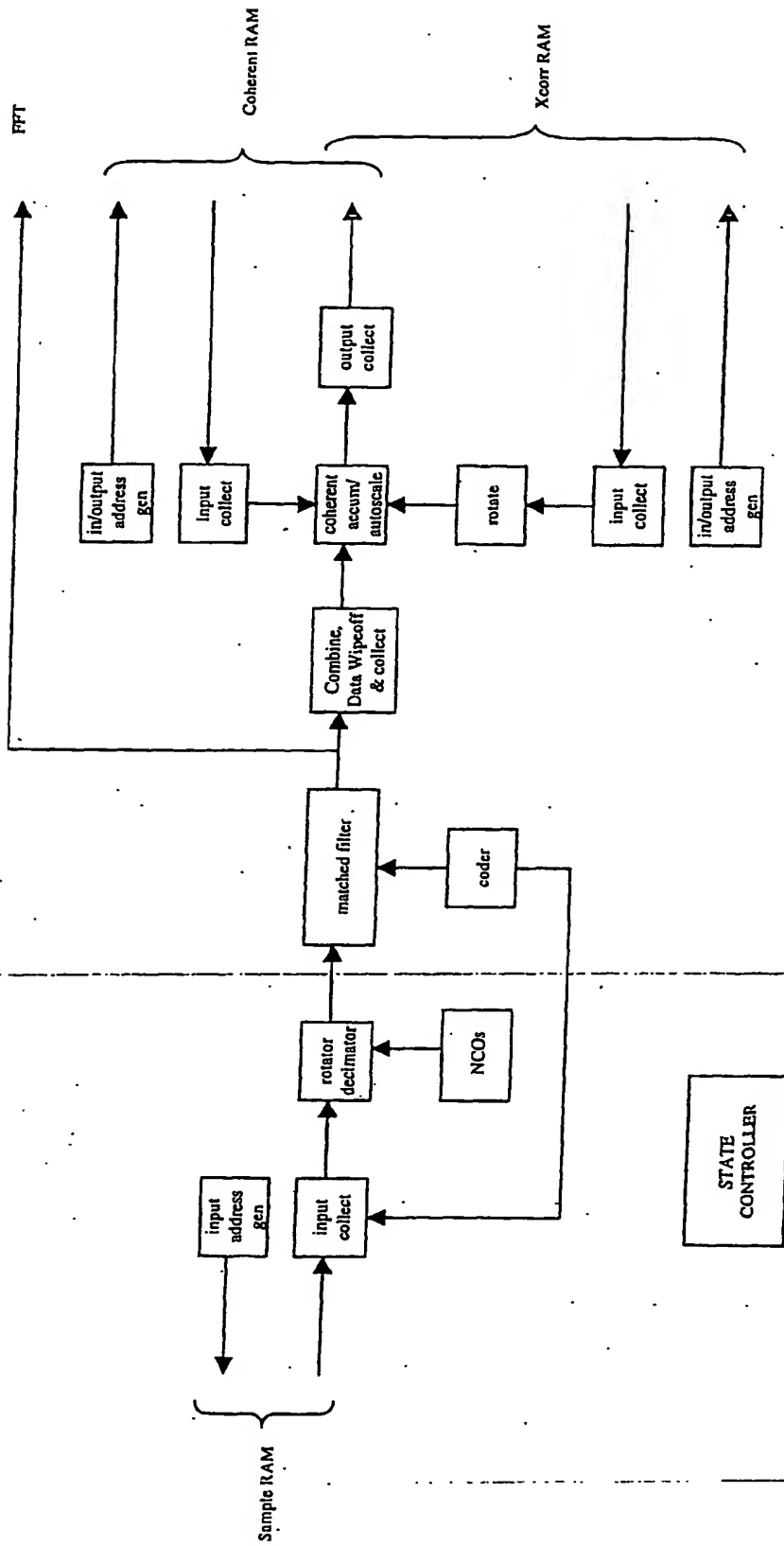


FIG. 9
PART II
signal processor subsystem flow

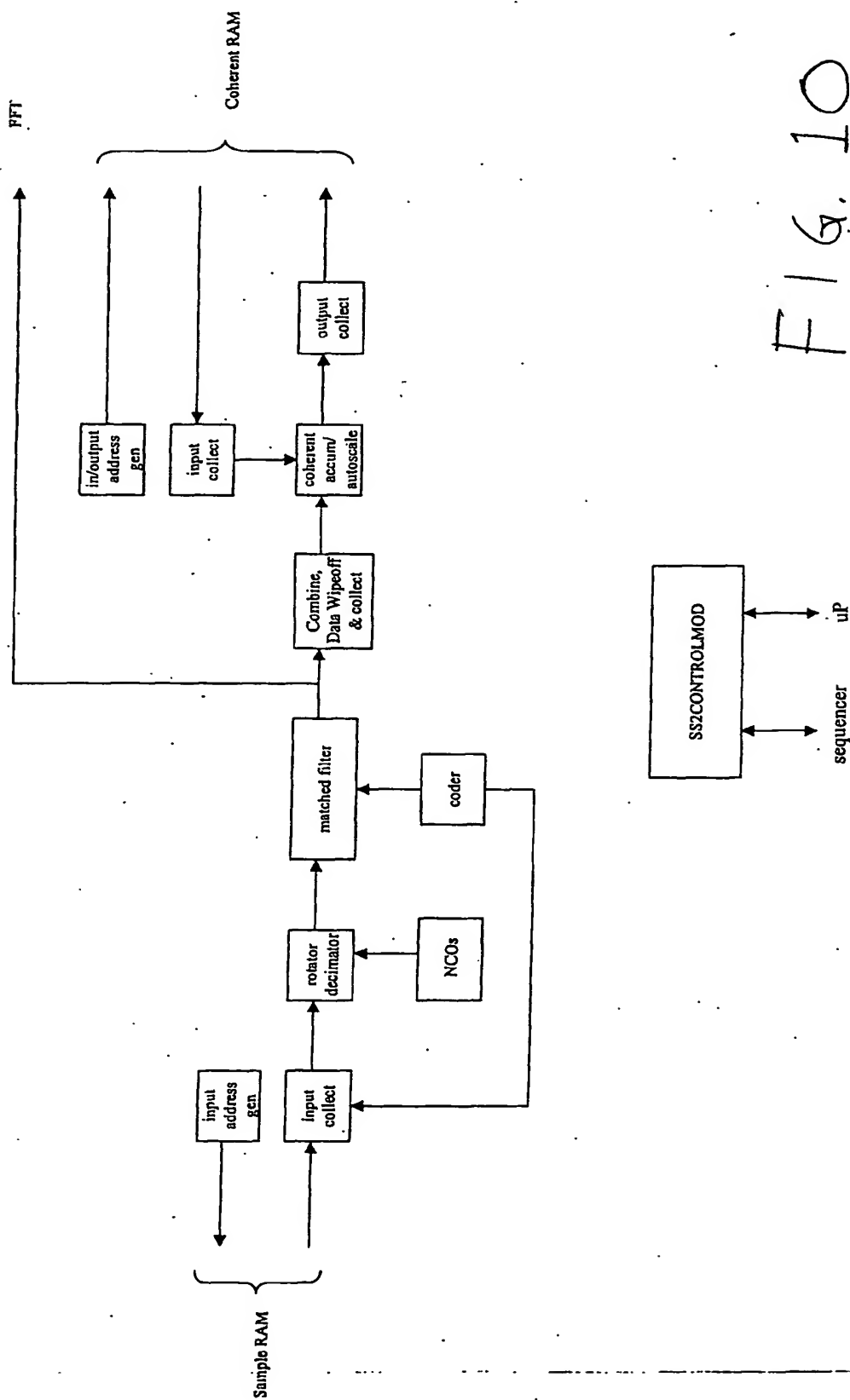
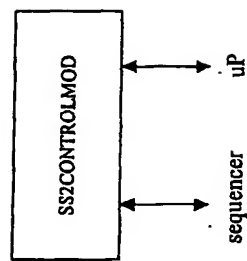
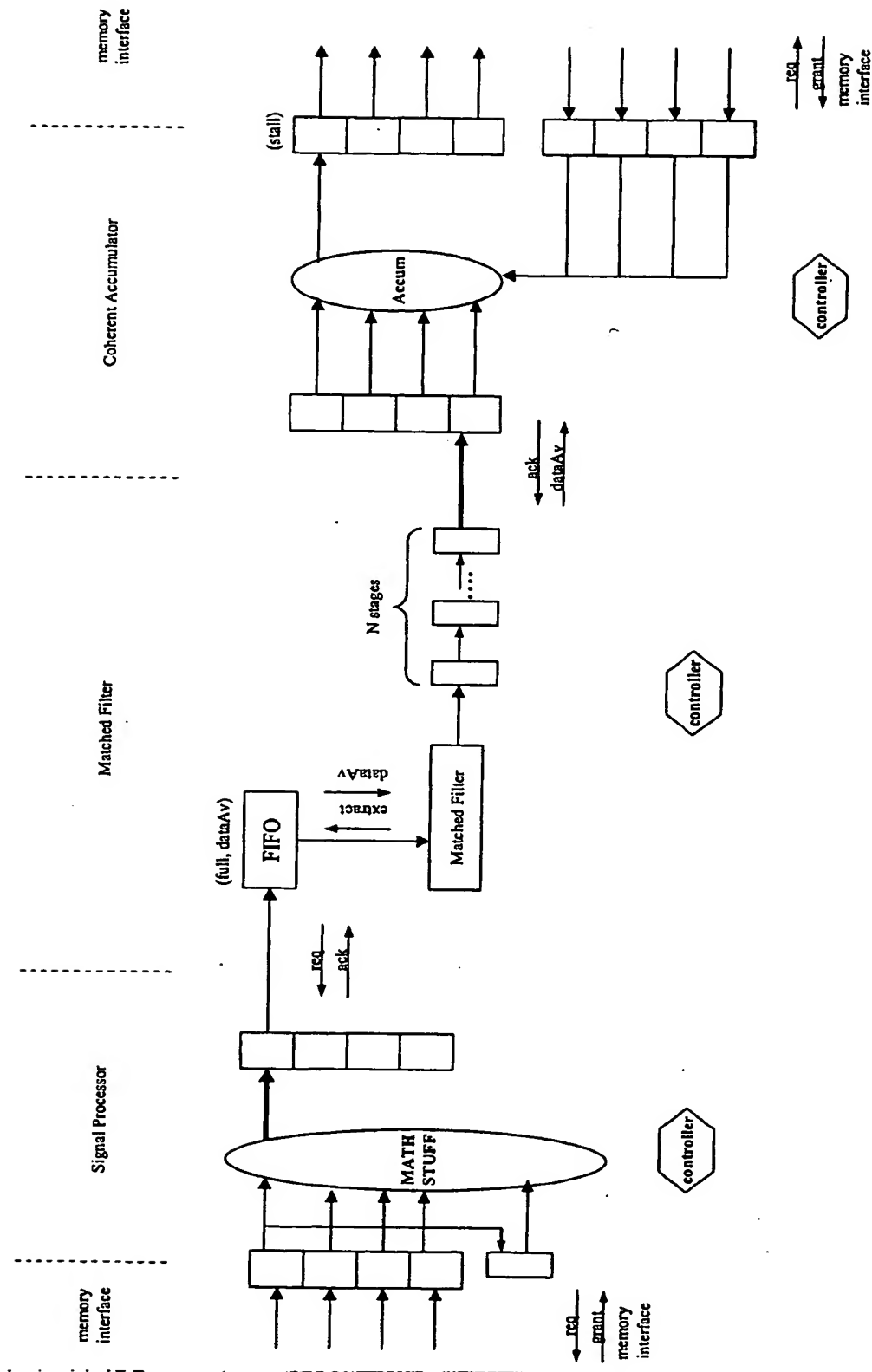


FIG. 10
PART II

signal processor subsystem flow





subsystem 2 data flow control

FIG. 11 PART II

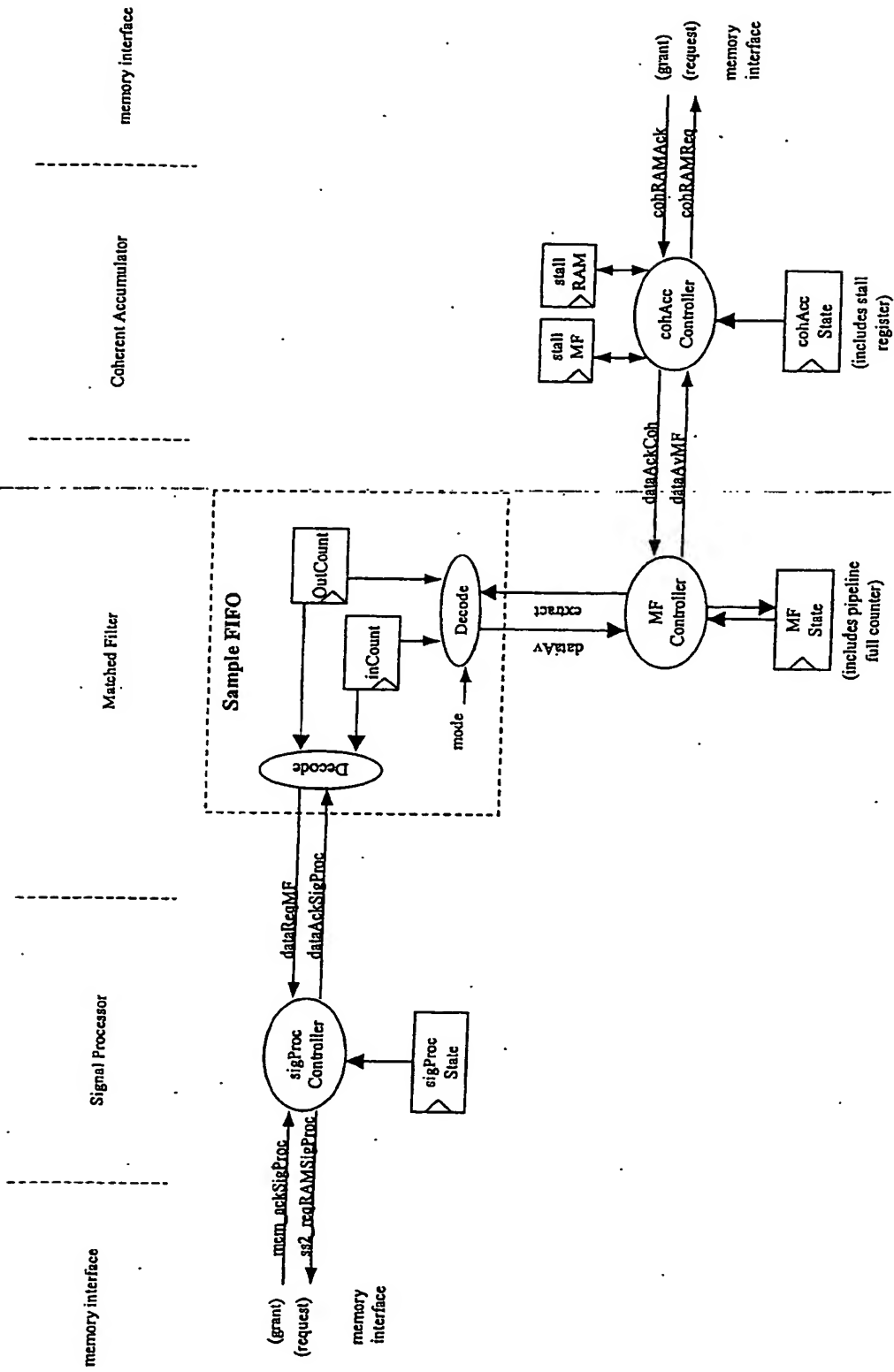
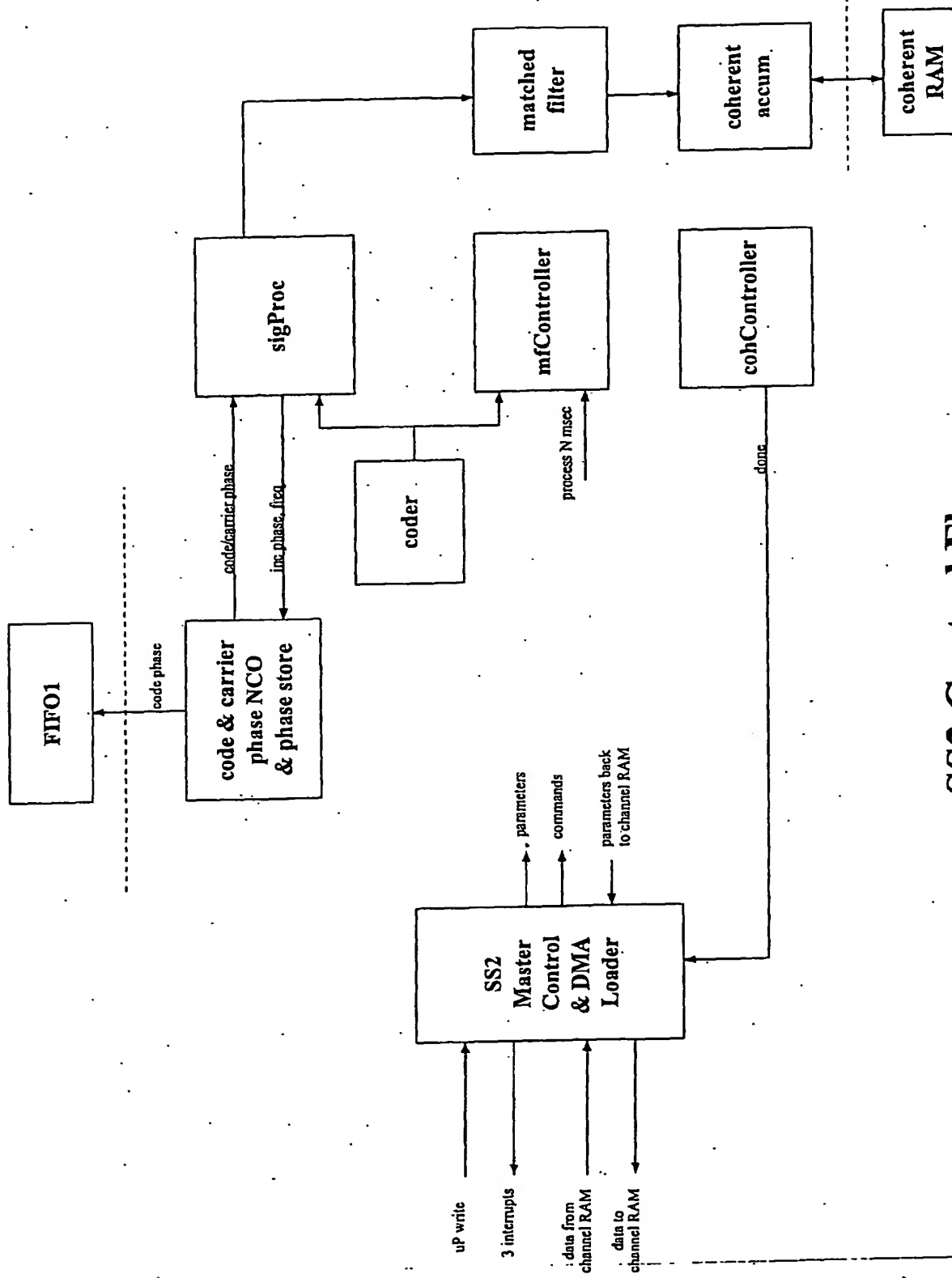


FIG. 12
PART II

subsystem 2 data flow control



SS2 Control Flow

Fig. 13 PART II

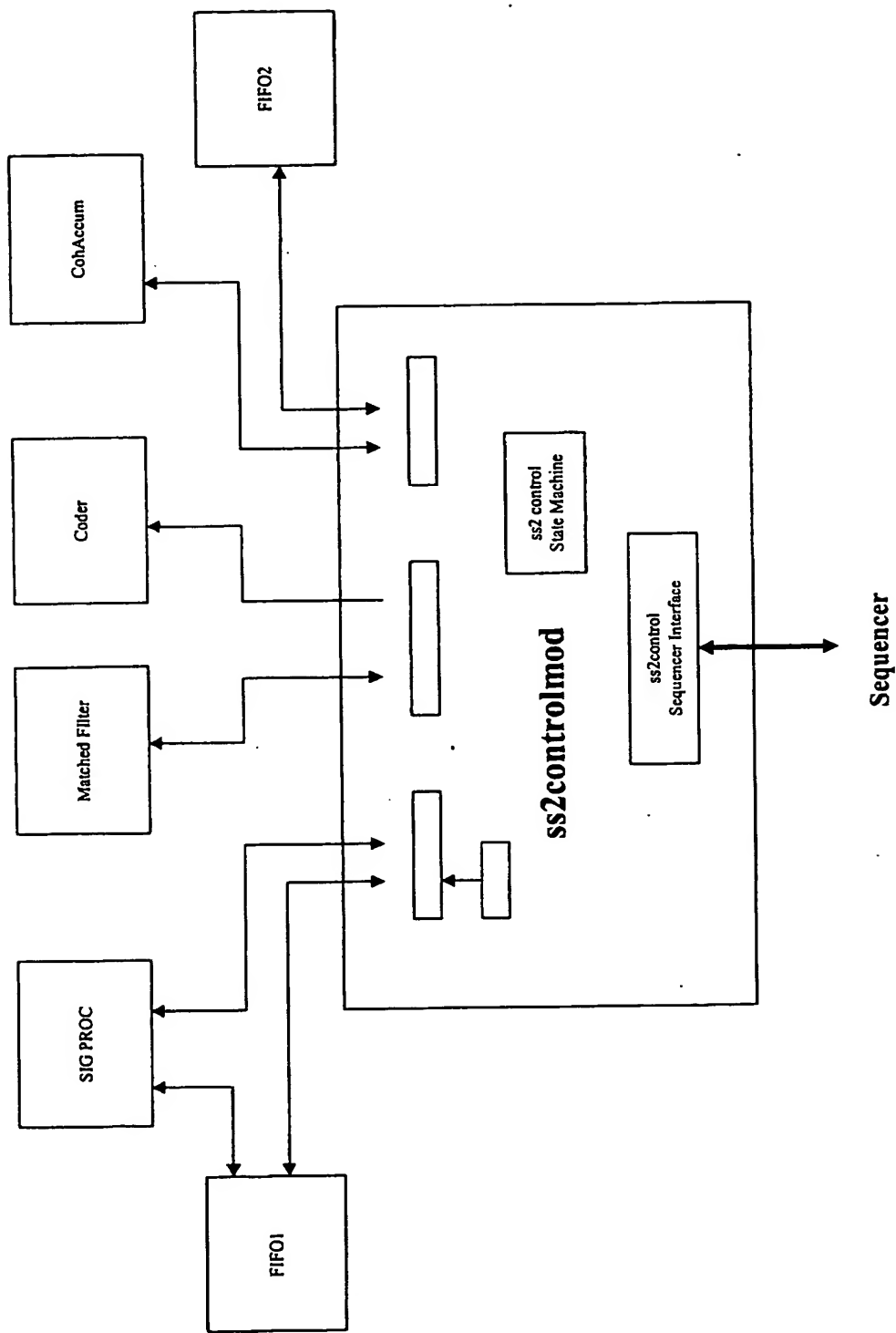
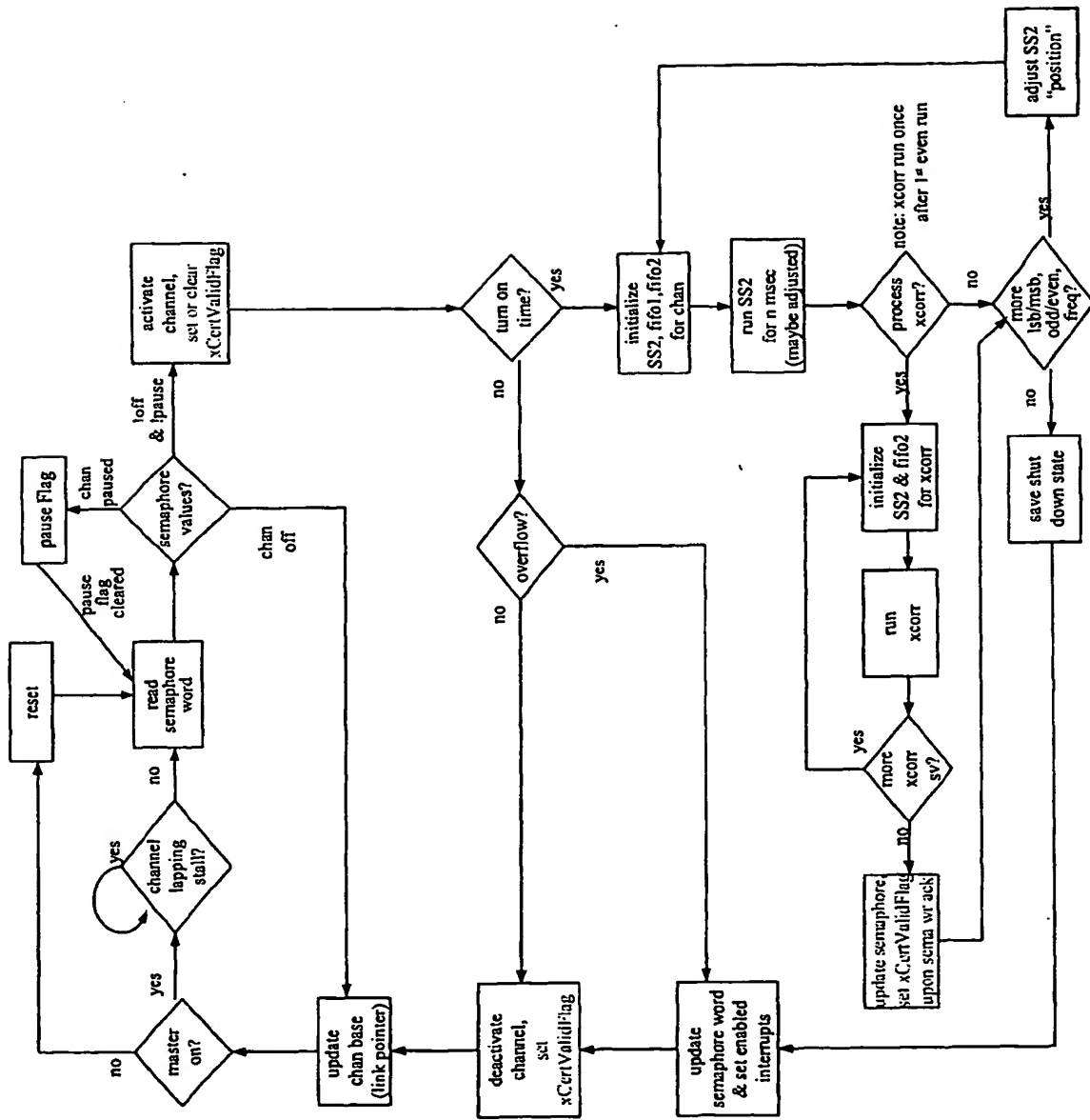


FIG. 14
PART II

ss2controlmod interface



PART II

SS2 master controller flow

F1615

ss3inmod

ffimod

ncamod

ss3outmod

Peak
sortedmod

ss3controlmod

PART II

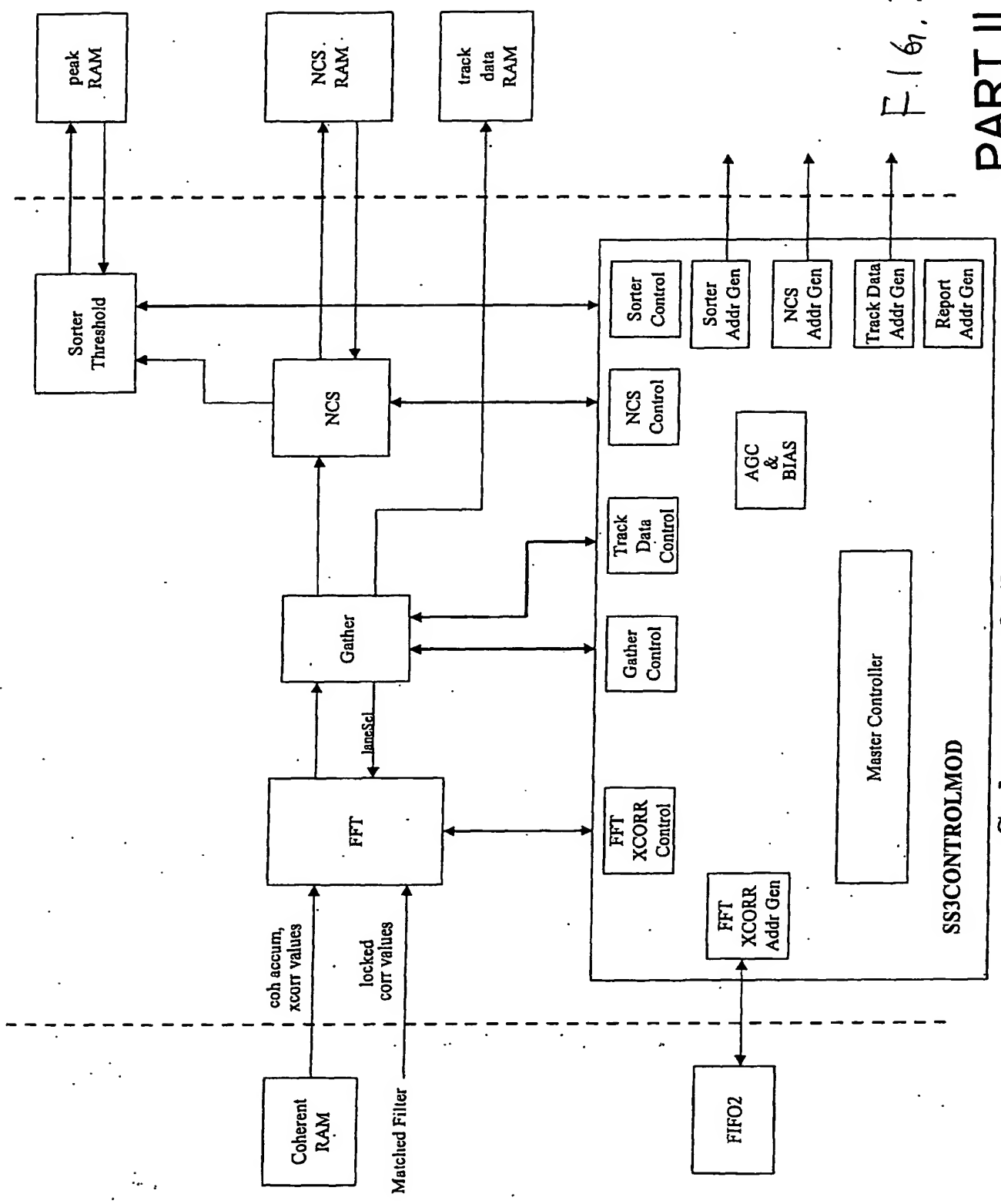
FIG. 16

fft subsystem partitioning

File 17

PART II

Subsystem 3 Functional Flow



SS3CONTROLMOD

Master Controller

FFTXCORR
Addr Gen

FIFO2

FFTXCORR
Control

Gather
Control

Track
Data
Control

NCS
Control

Sorter
Control

AGC &
BIAS

Sorter
Addr Gen

NCS
Addr Gen

Track Data
Addr Gen

Report
Addr Gen

Coherent
RAM

FFT

Gather

NCS

Sorter
Threshold

peak
RAM

NCS
RAM

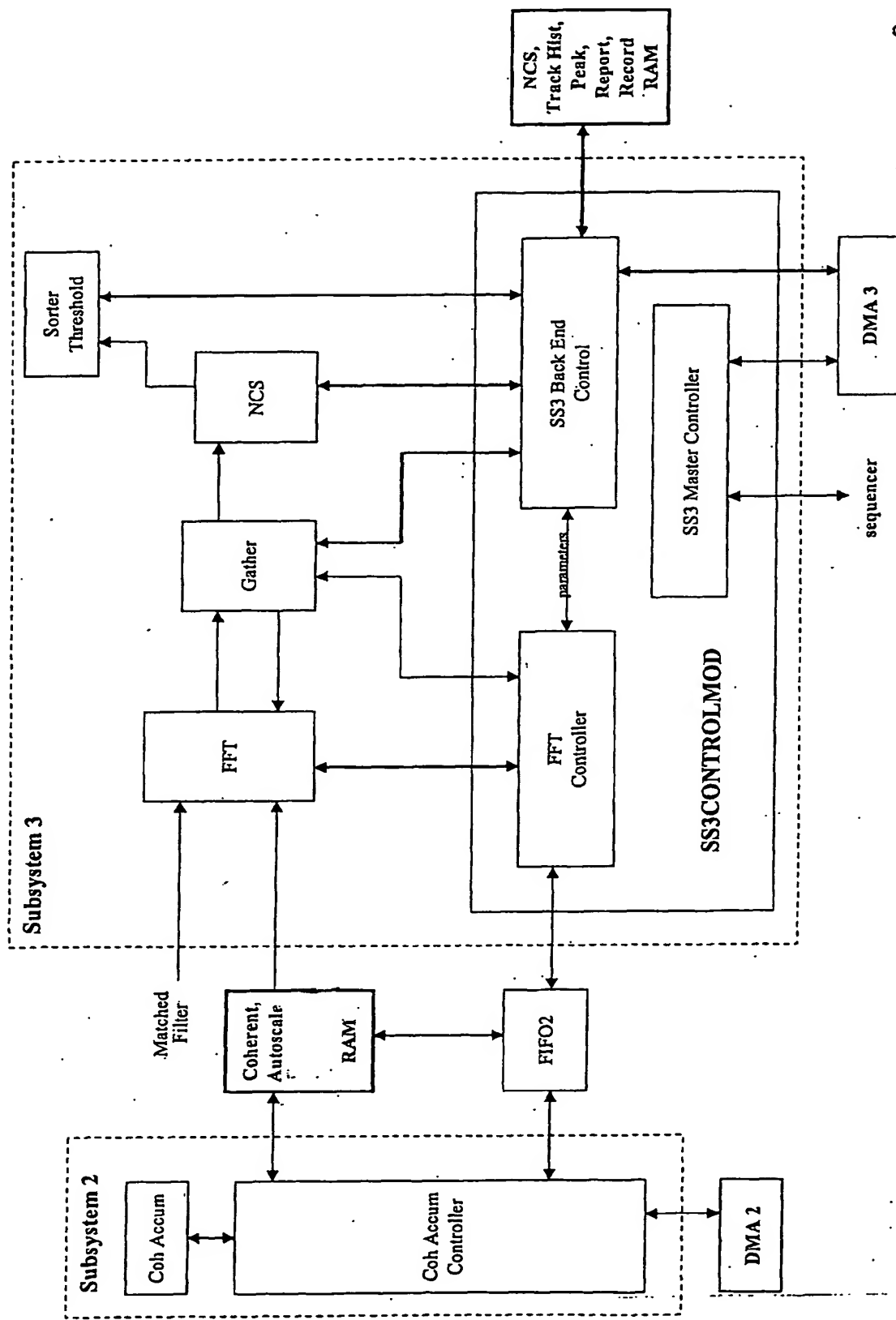
track
data
RAM

coh accum,
xcrr values

locked
corr values

Matched Filter

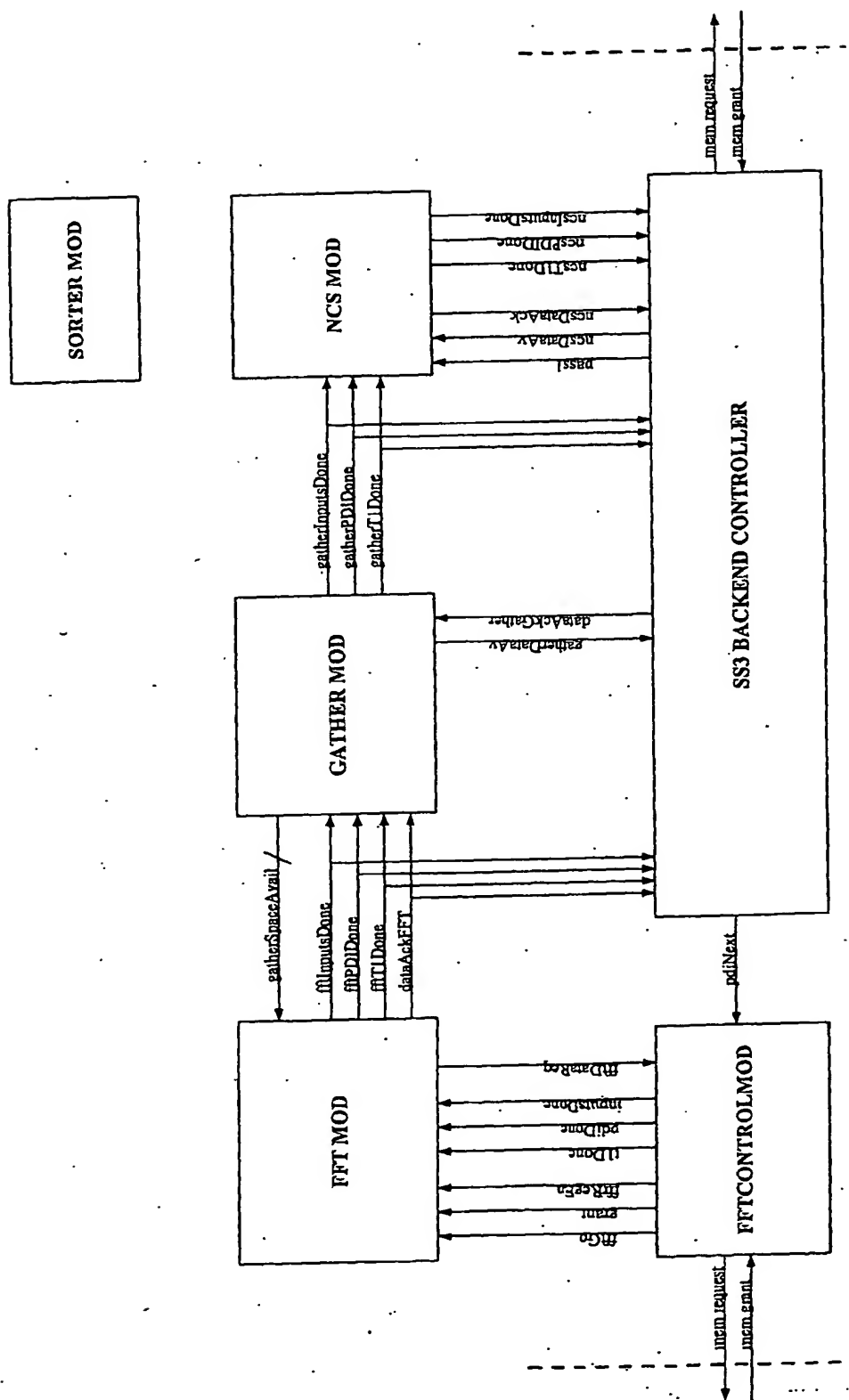
laneSel



8714

PART II

Subsystem 3 Functional Flow (2)



6719

Subsystem 3 Control Flow

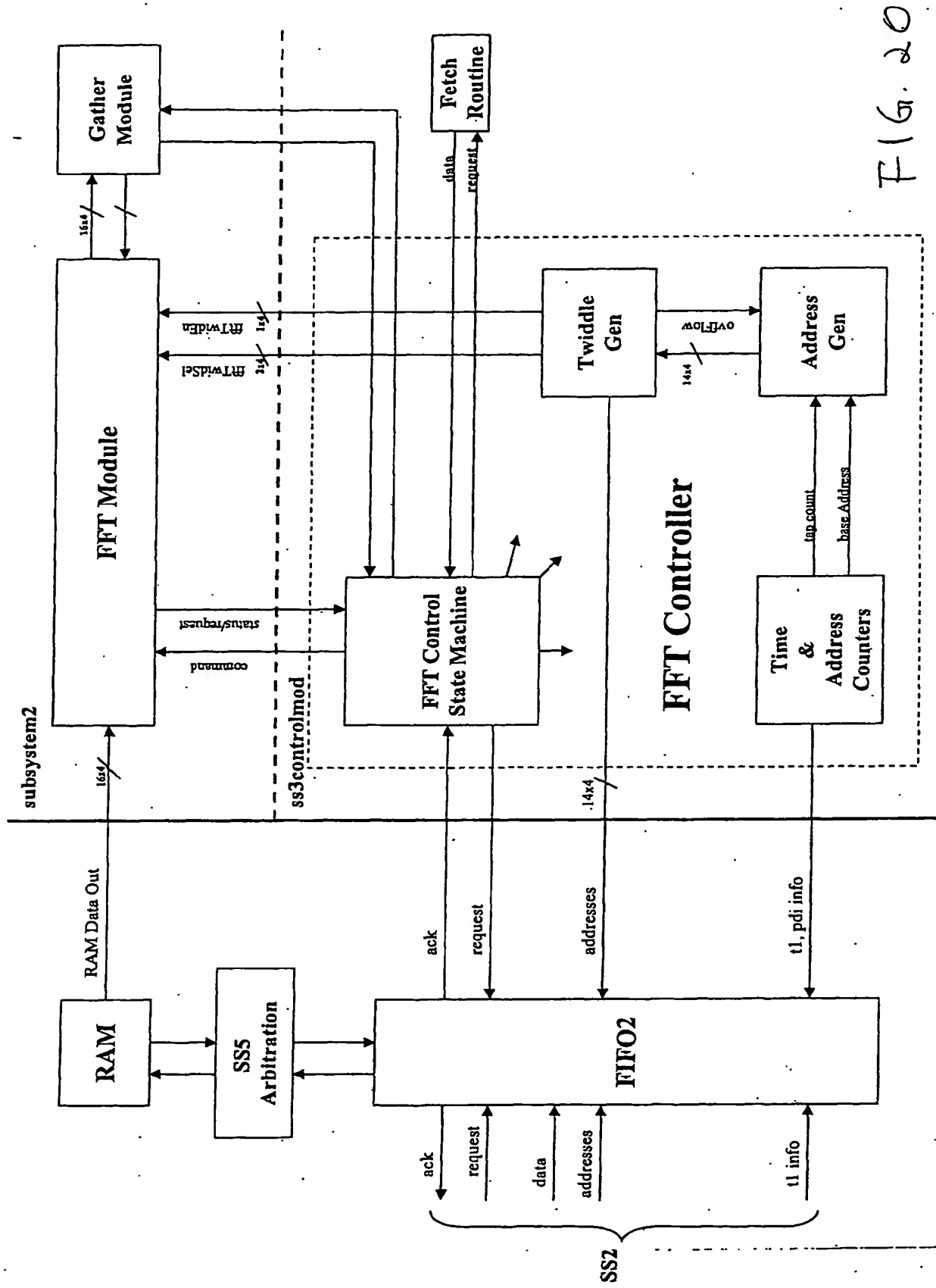


FIG. 20

FFT Controller Flow

PART II

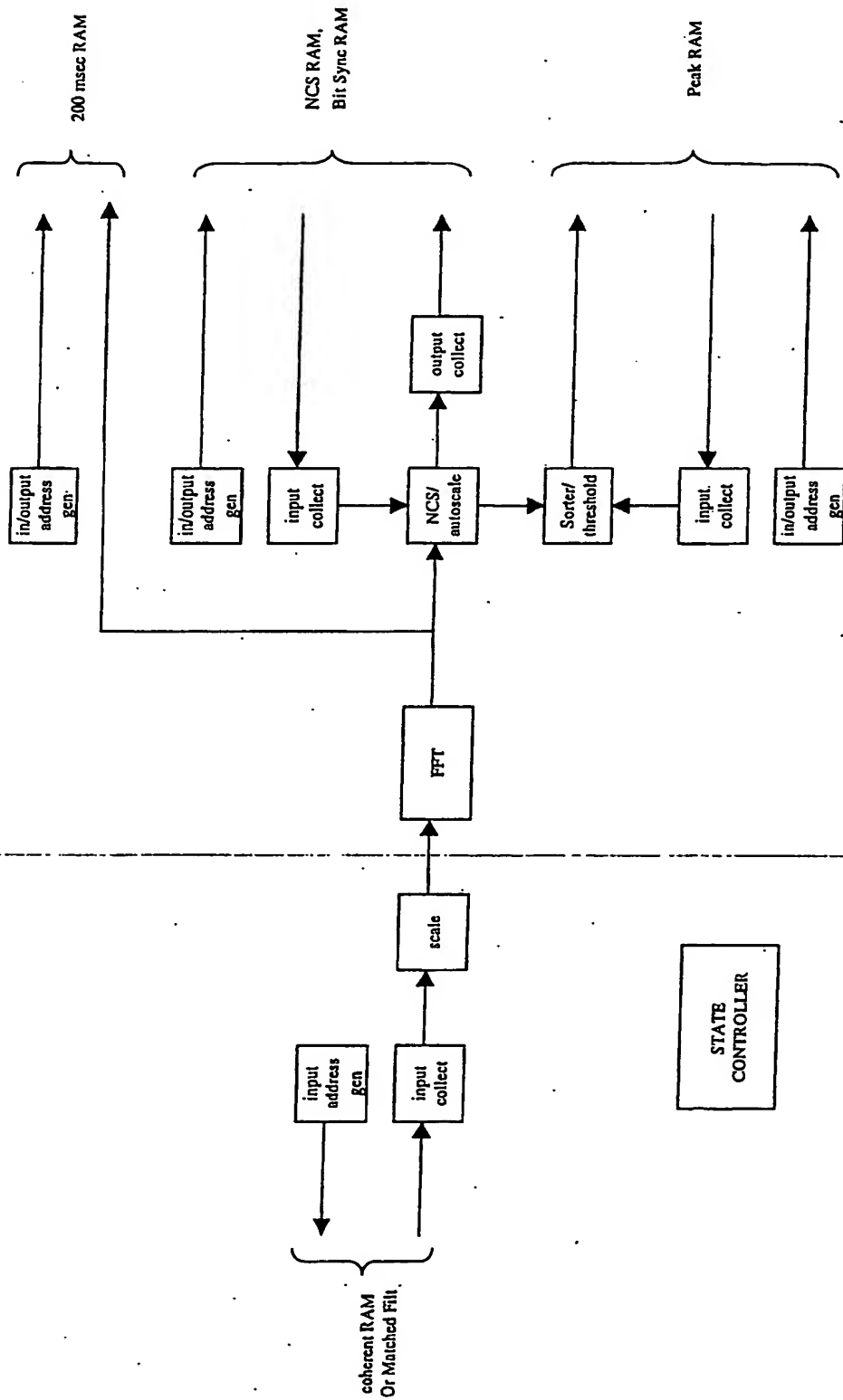


Fig. 21
PART II

fft subsystem flow

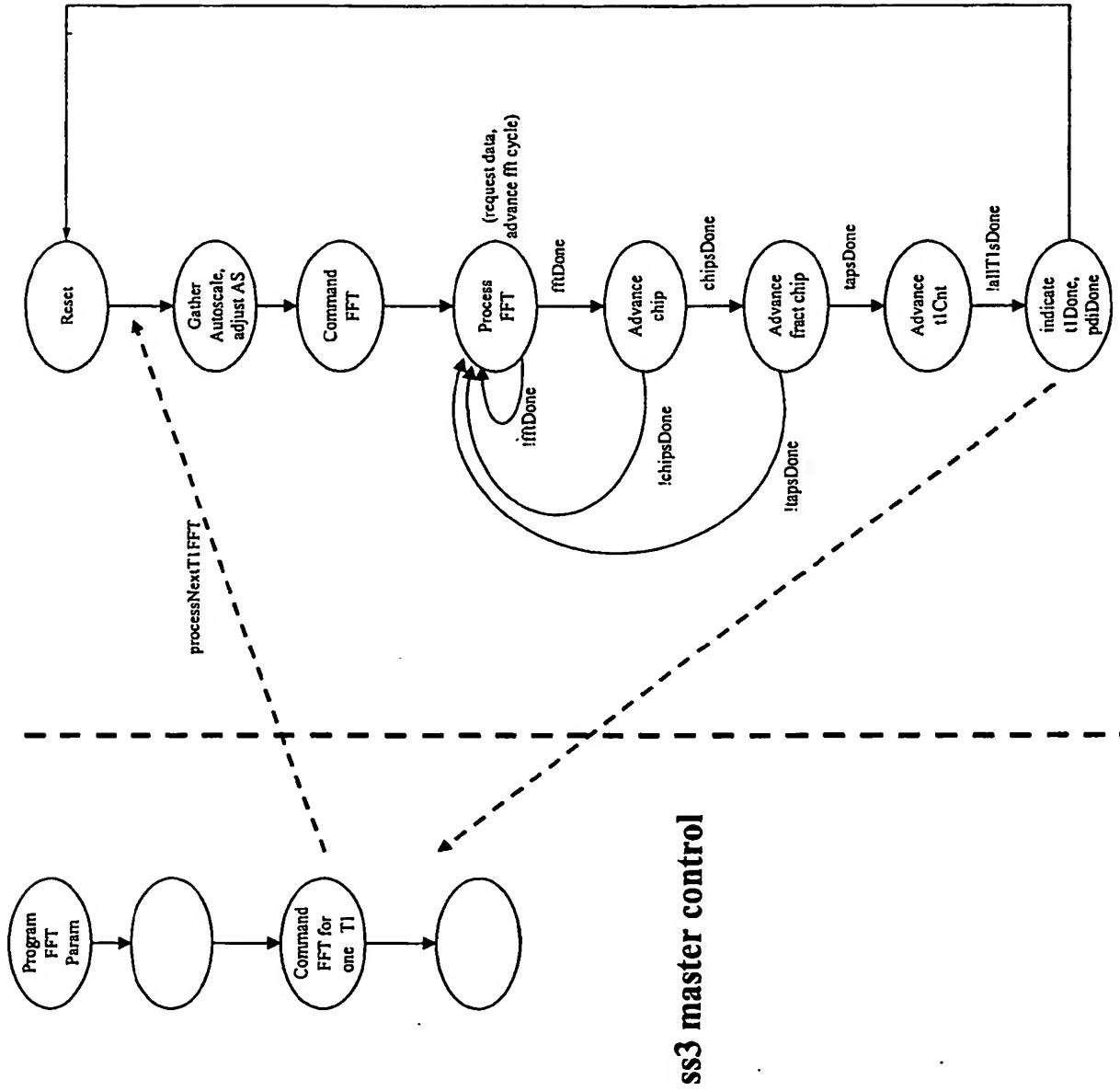
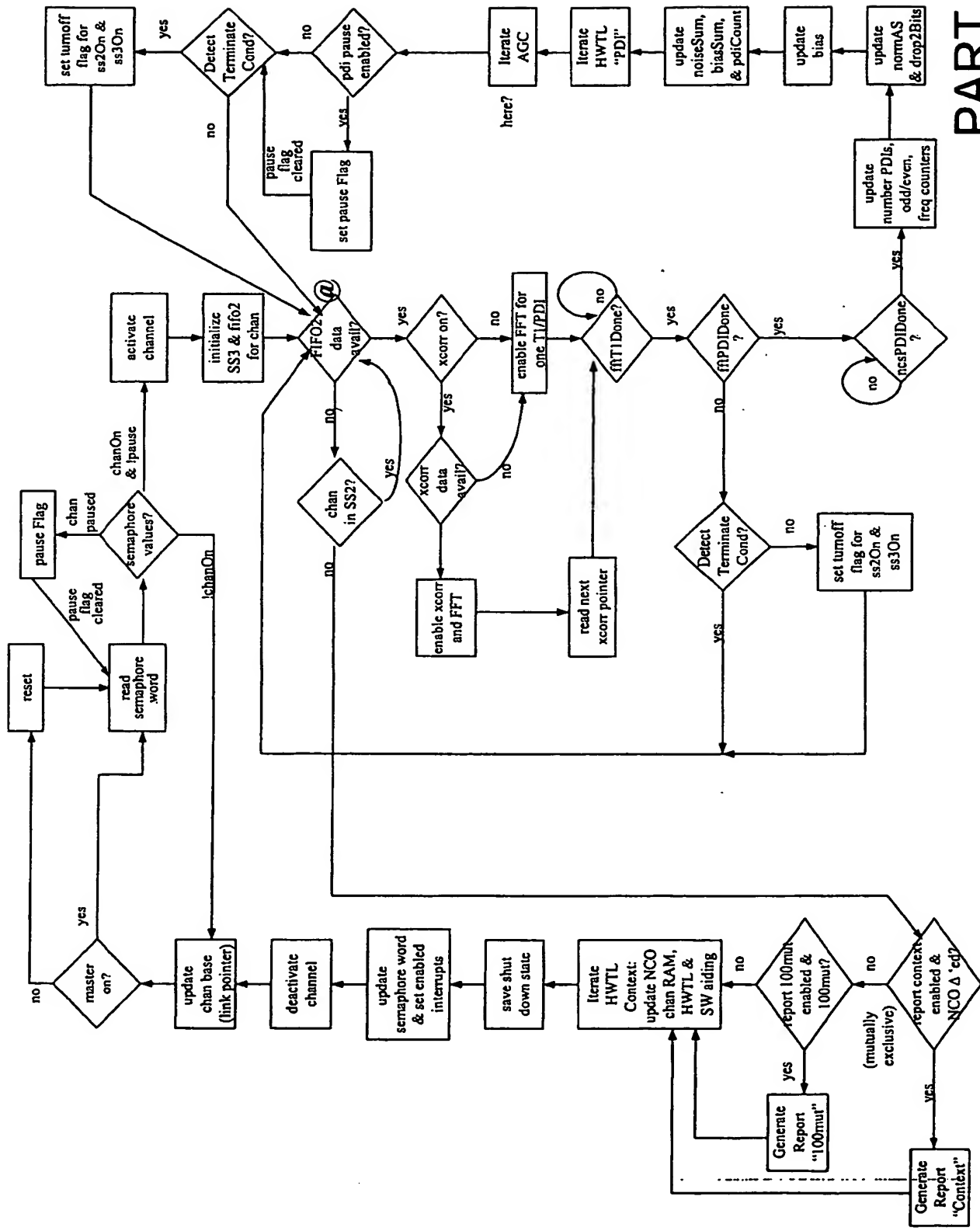


Fig. 22

PART II

FFT controller

FFT Controller Flow

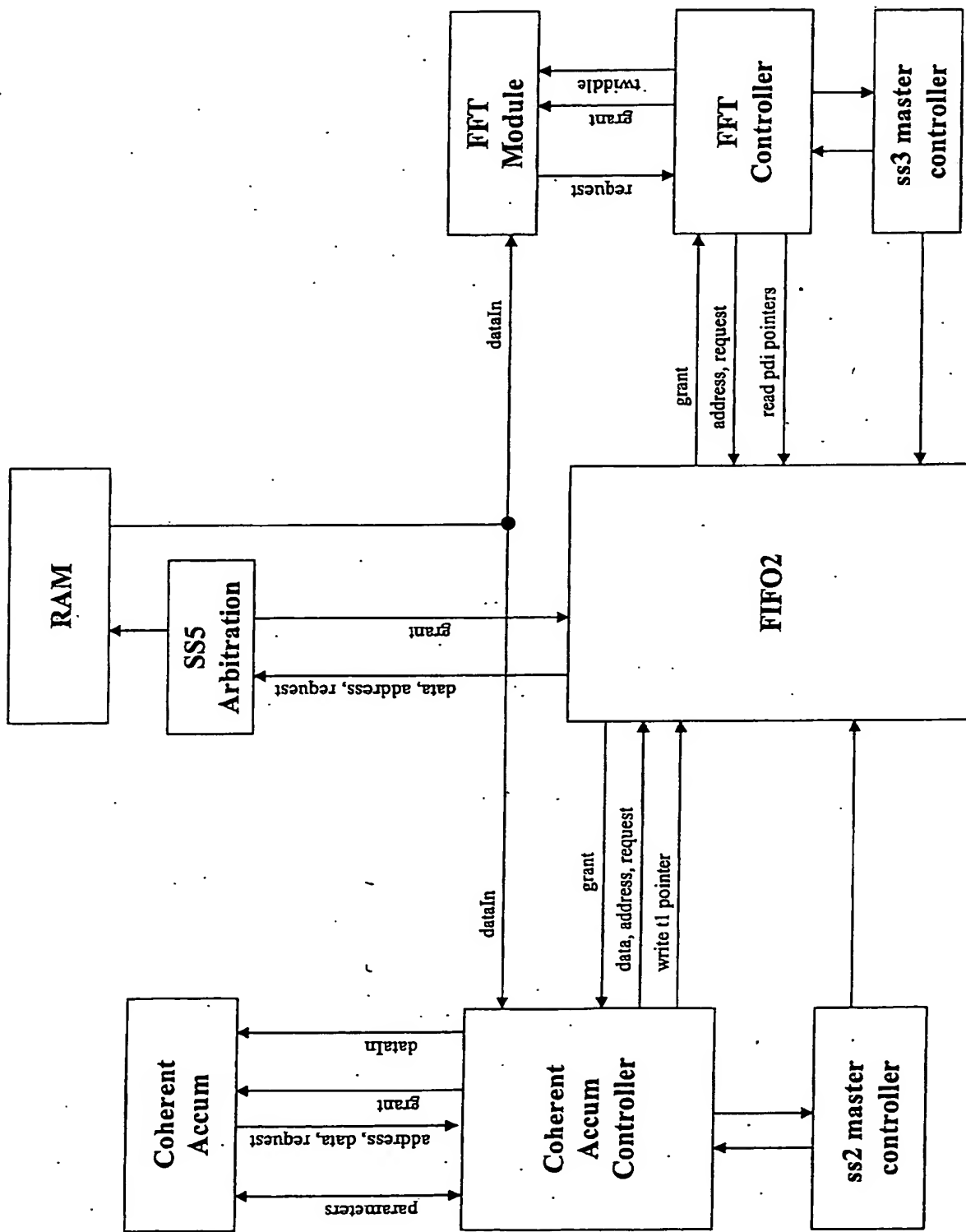


note: @ in locked mode use direct data
available signal from SS2

SS3 master controller flow

F16, 23

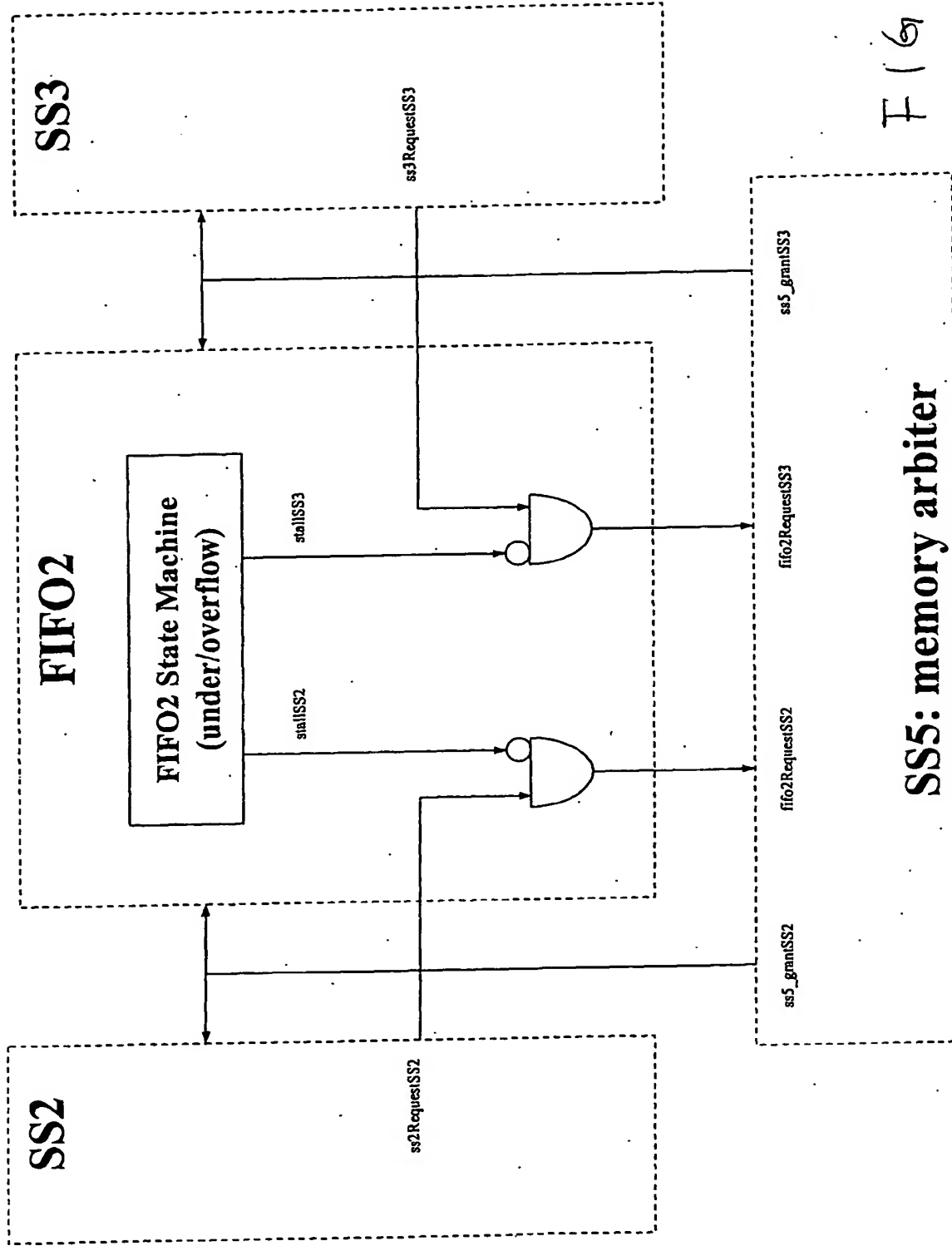
PART II



Subsystem2 / FIFO02 / Subsystem3 Flow

PART II

FIG. 24



F169.25

PART II

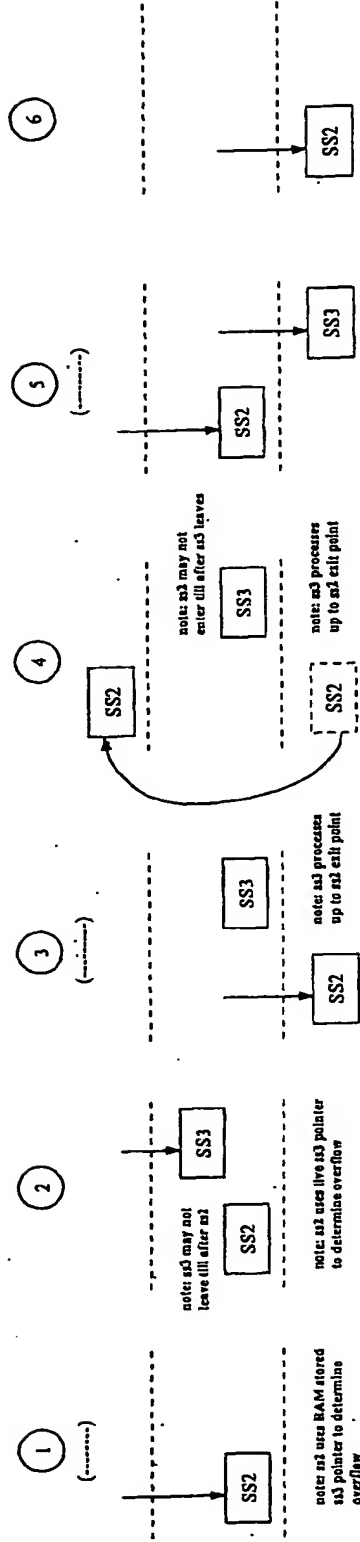


Fig. 2.6

PART II

SS2/SS3 FIFO2 Interactions

sequpmod

seqstatemod

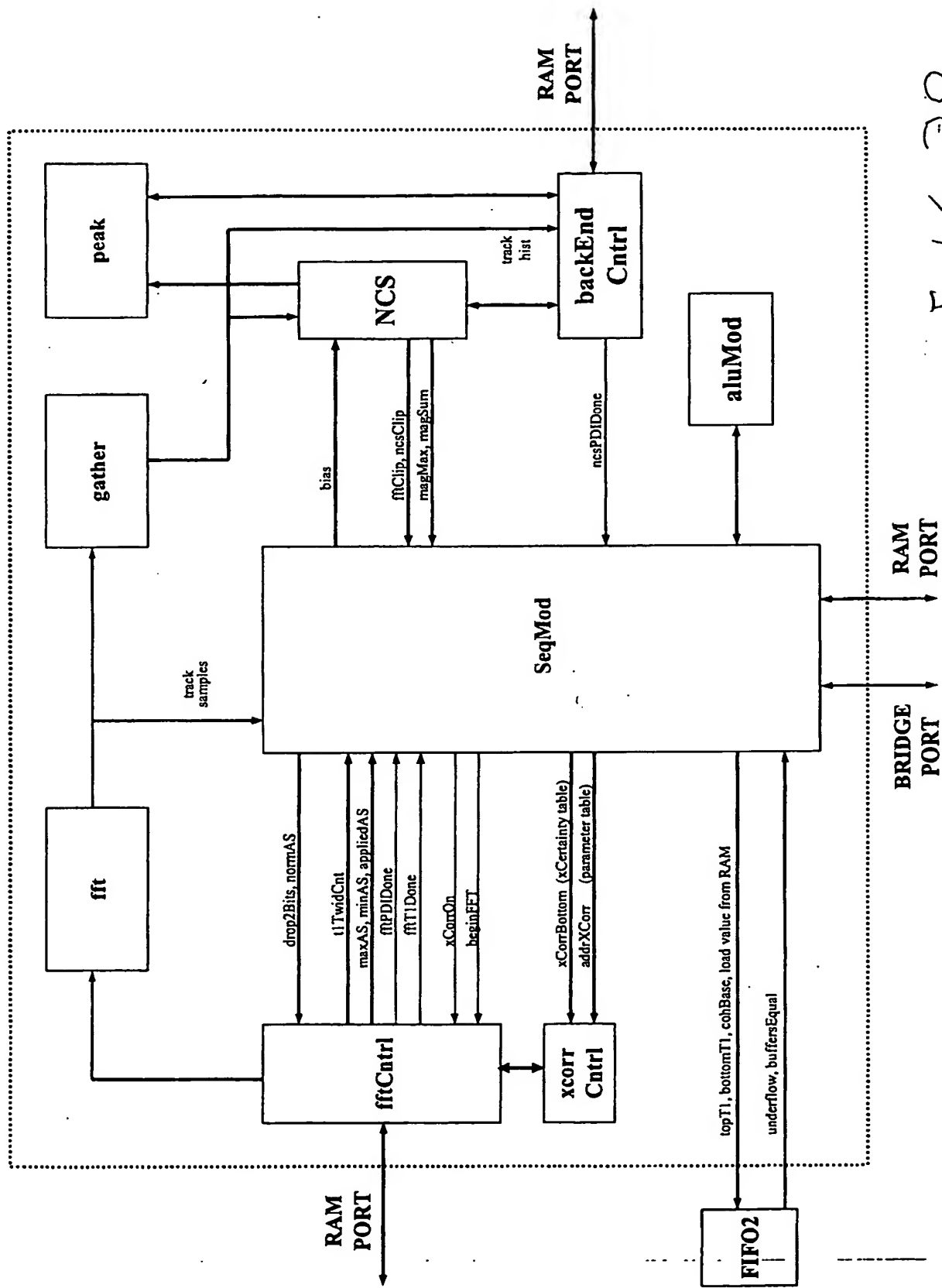
sequnmod

seqintrfmod

PART II

F167.27

sequencer subsystem partitioning



F 16, 28

Sequencer Module Interface

PART II

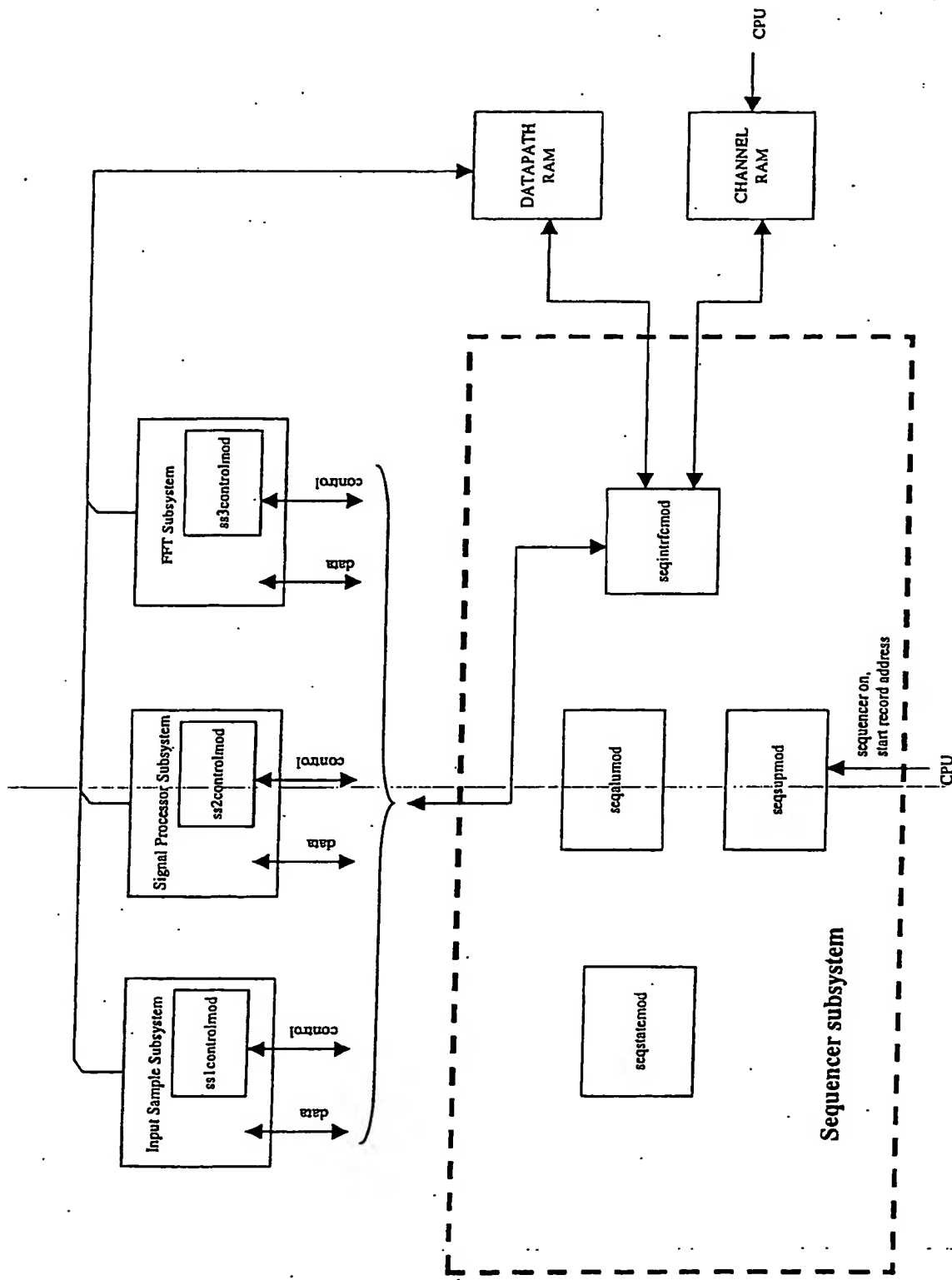
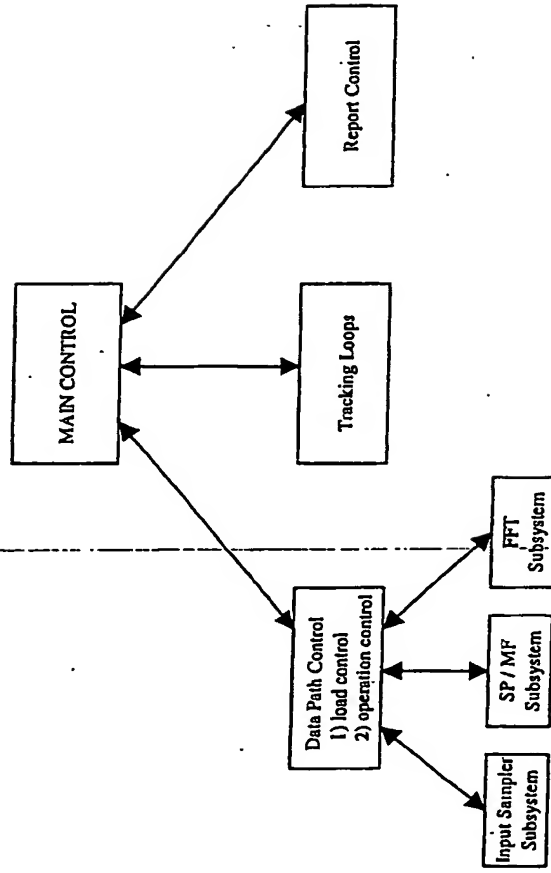


FIG. 29

sequencer subsystem flow

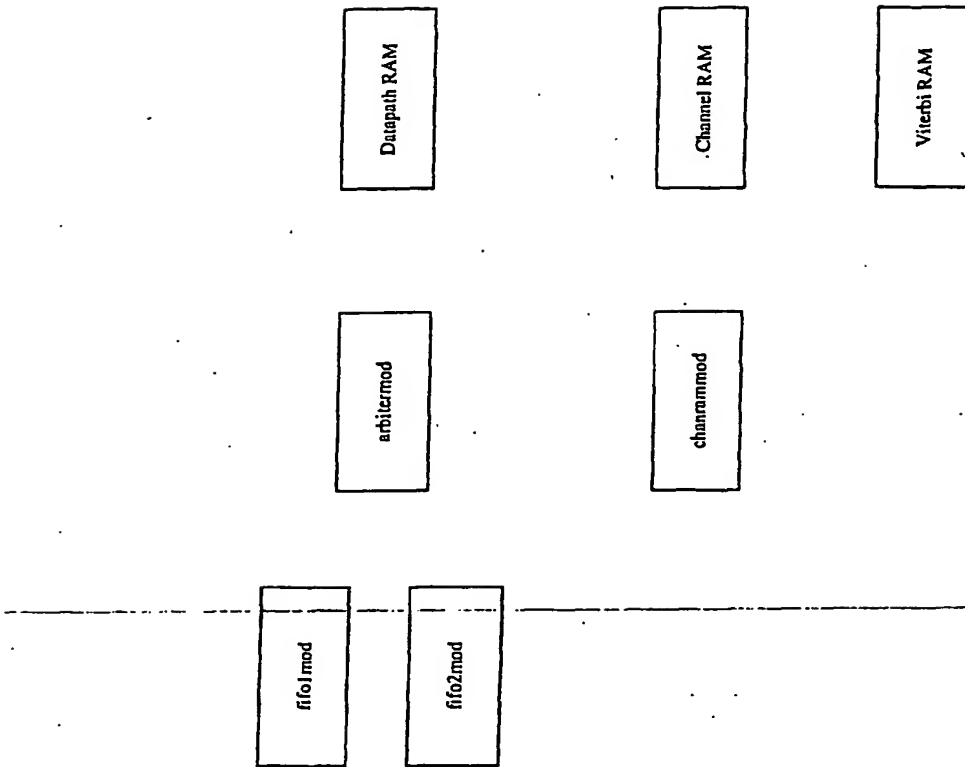
PART II



PART II

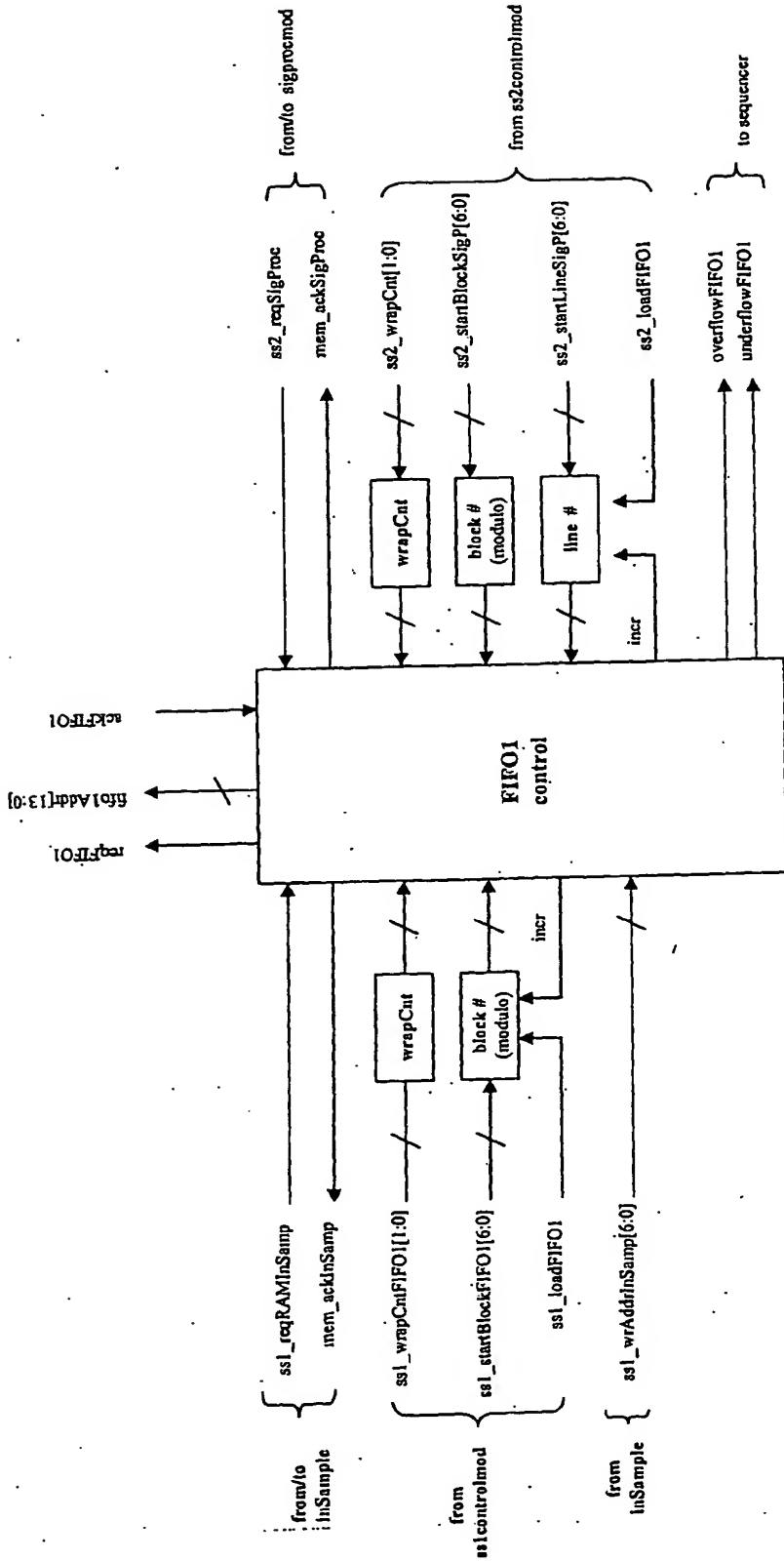
Fig. 30

sequencer state machine flow



memory subsystem

Fig. 31
PART II



Note:
 sample rate 16 samp/chip
 4 samp/chip
 128 lines / block
 line 2 chips/line
 8 chips/line
 mSec 512 lines/mSec
 128 lines/mSec

Fig. 32
 PART II

fifo1 structure

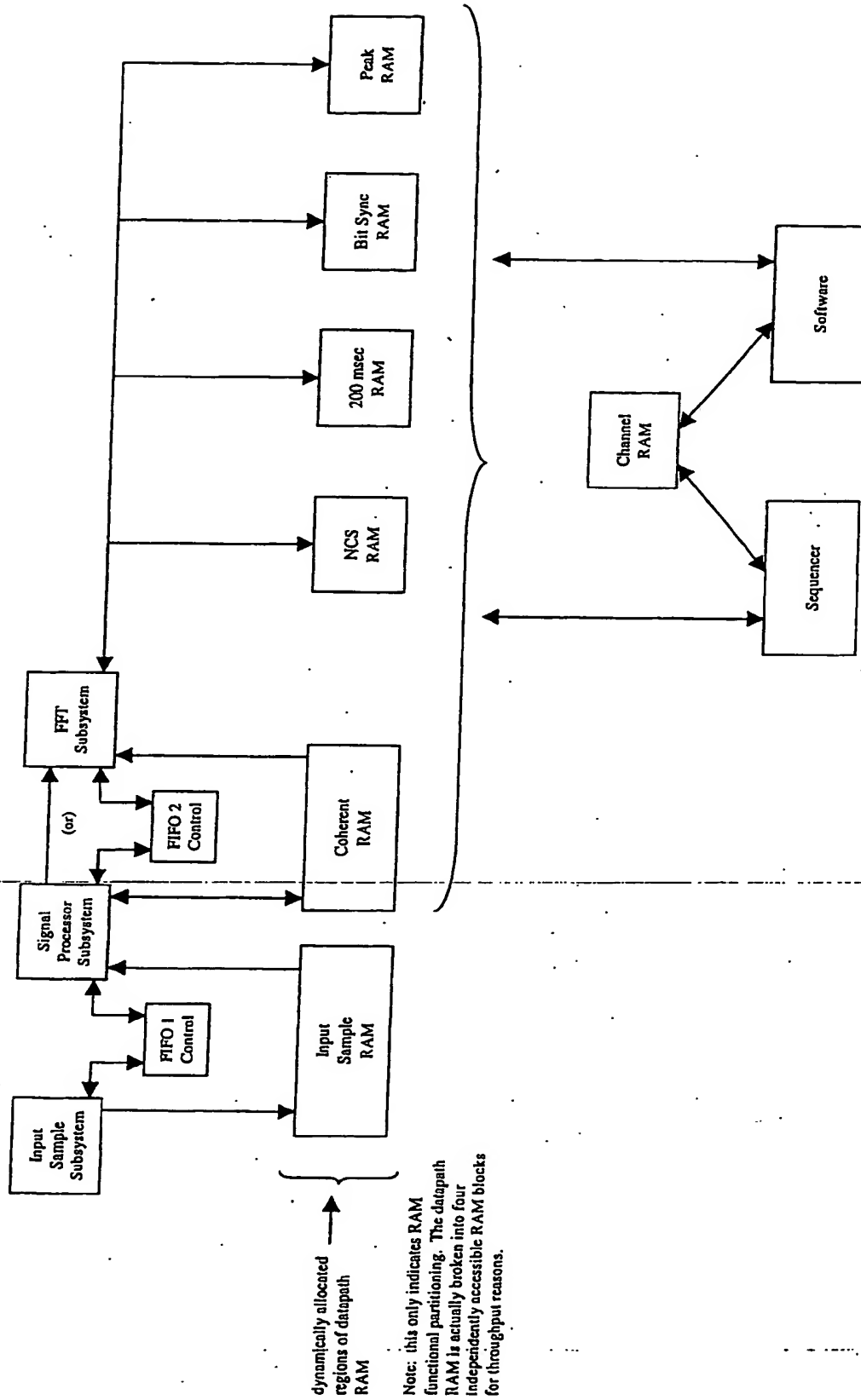
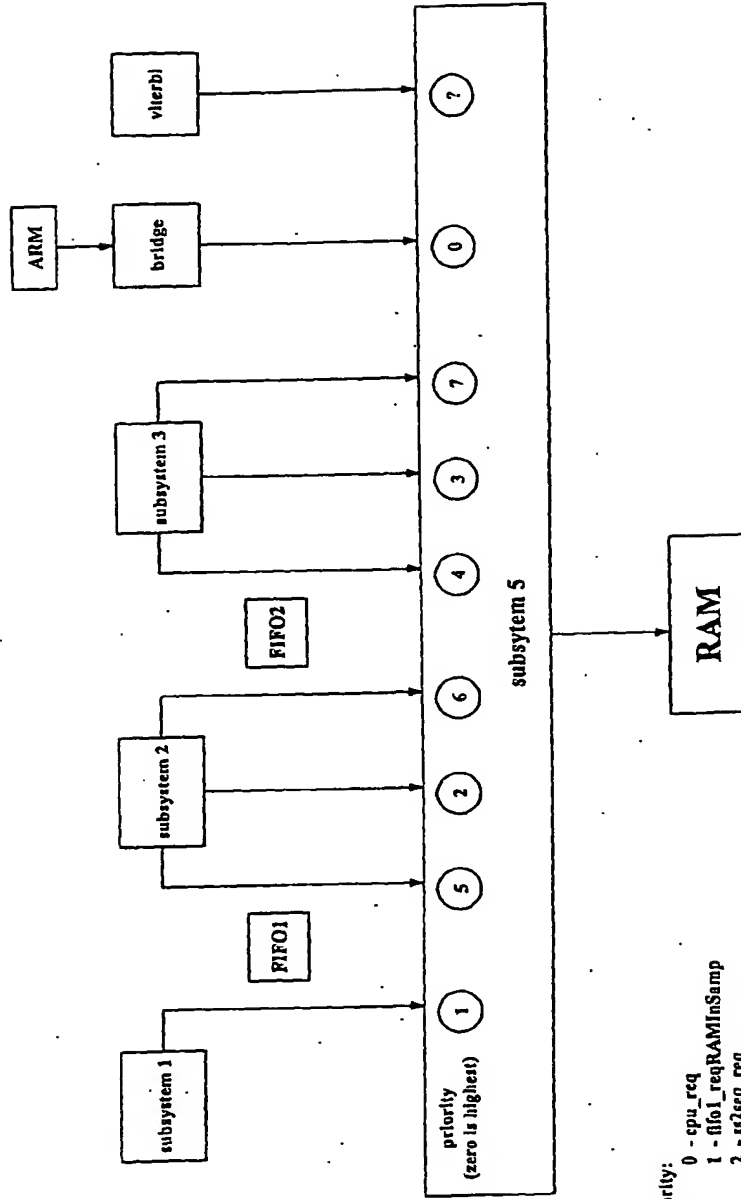


FIG. 33
PART II

Memory Data Path Flow



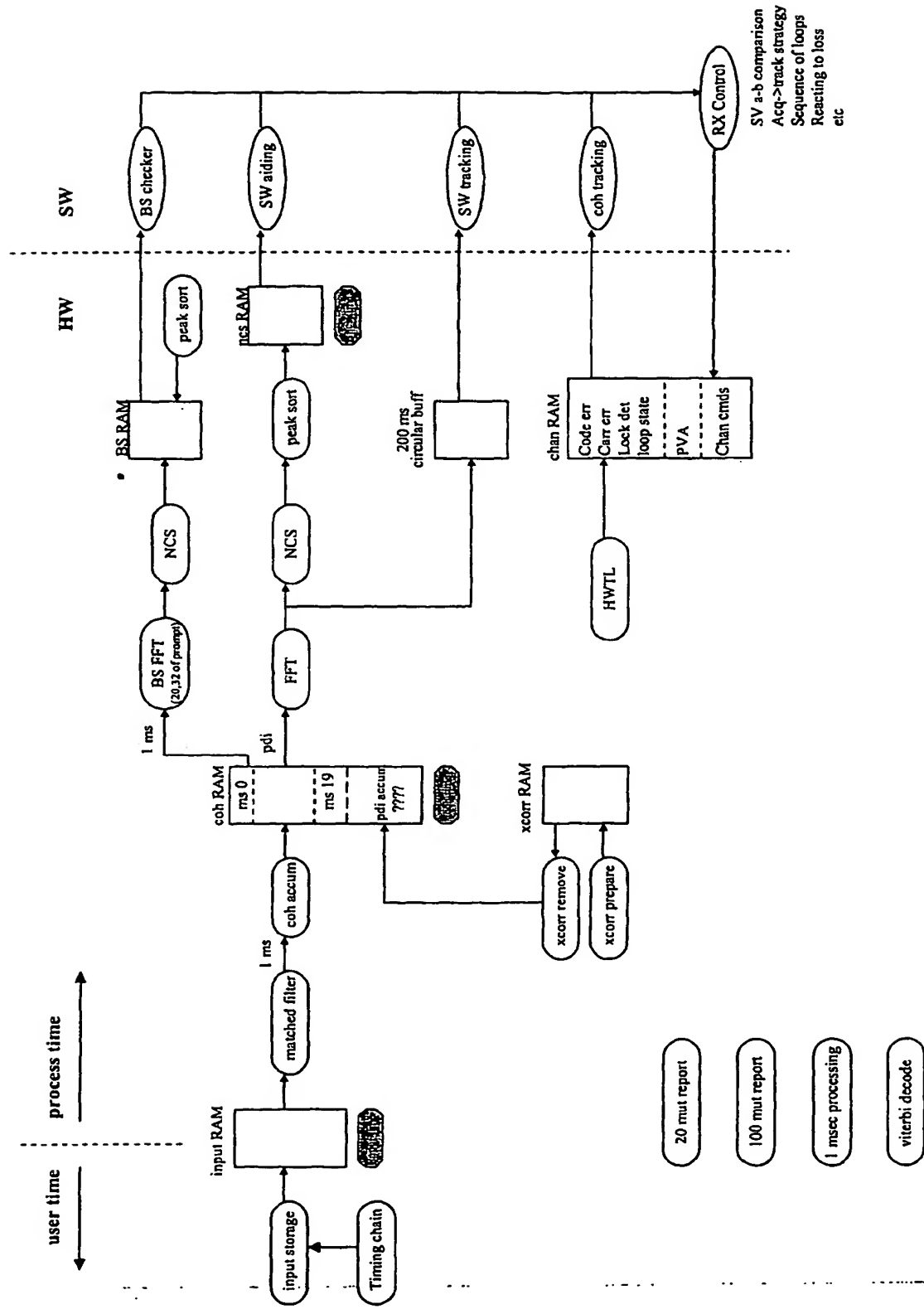
priority: ↑
higher

- 0 - cpu_req
- 1 - flfo1_reqRAMInSamp
- 2 - i2seq_req
- 3 - i3seq_req
- 4 - flfo2_reqRdFFT
- 5 - flfo1_reqRAMSigProc
- 6 - flfo2_cohRAMReq
- 7 - i3_beRAMReq
- 8 - viterbi ???

F16.34

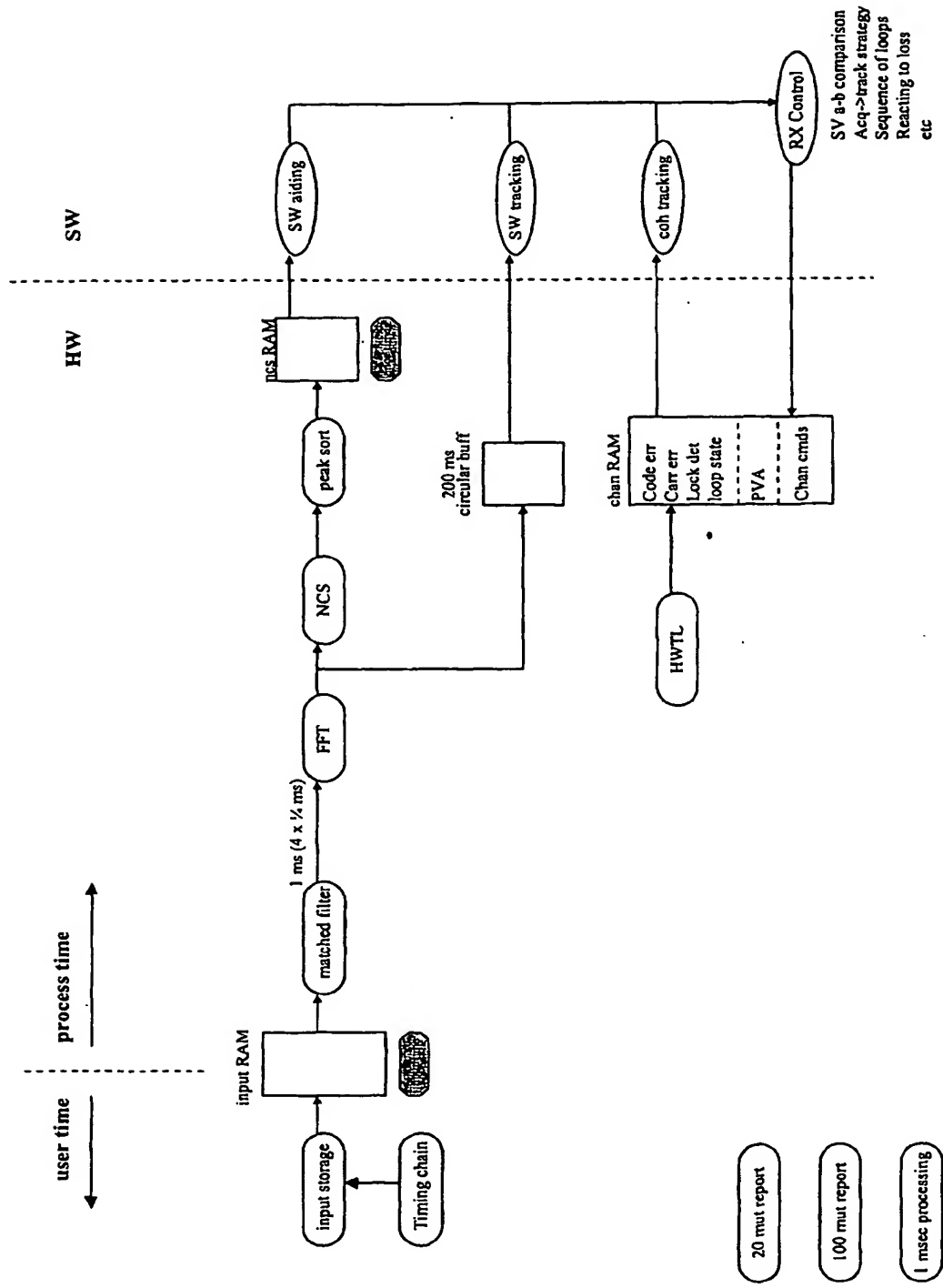
PART II

ss5 arbitration priority



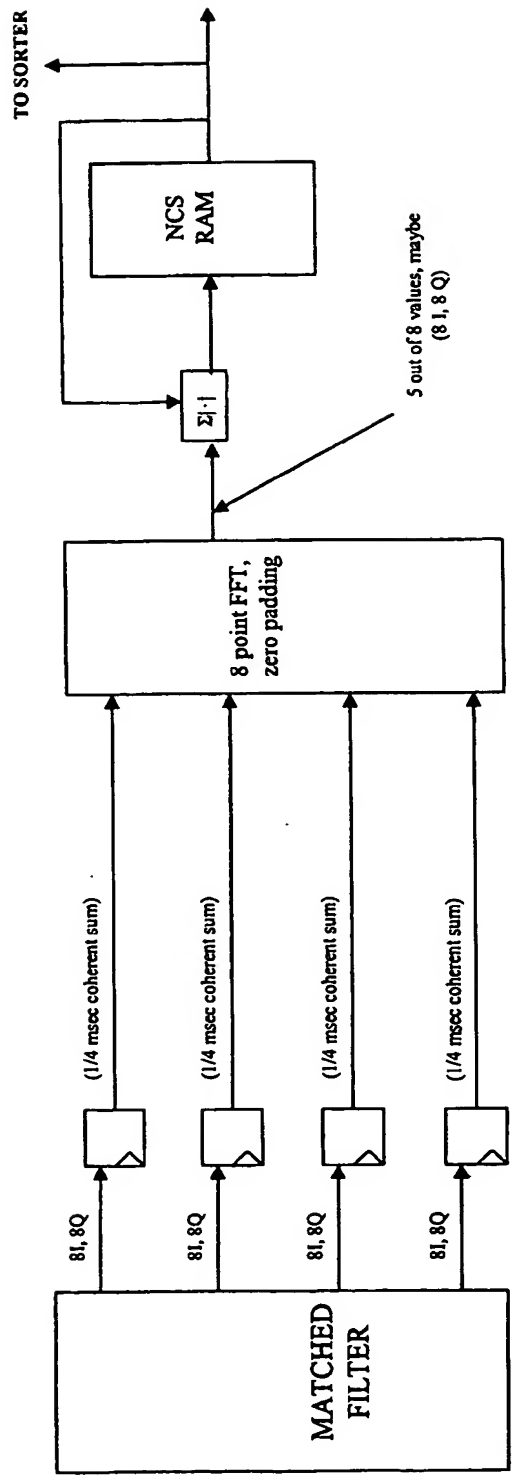
1 msec acq / 1/8 msec track mode process flow

Fig. 35 PART II

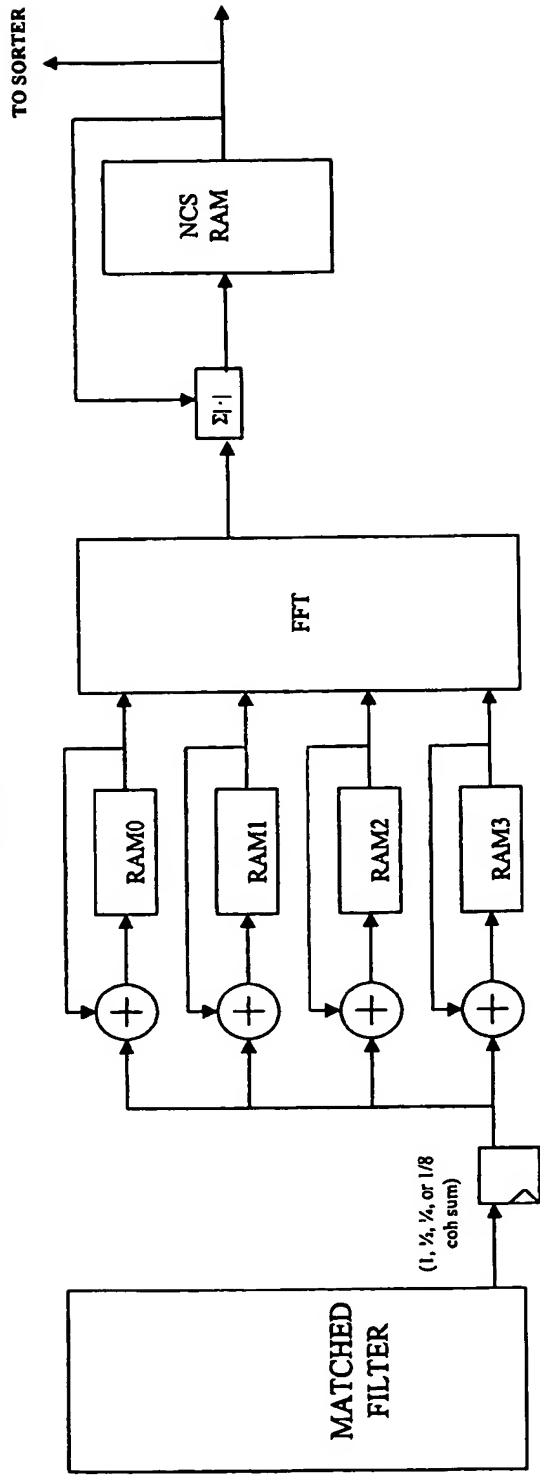


cold start acquisition mode process flow

F16.34 PART II



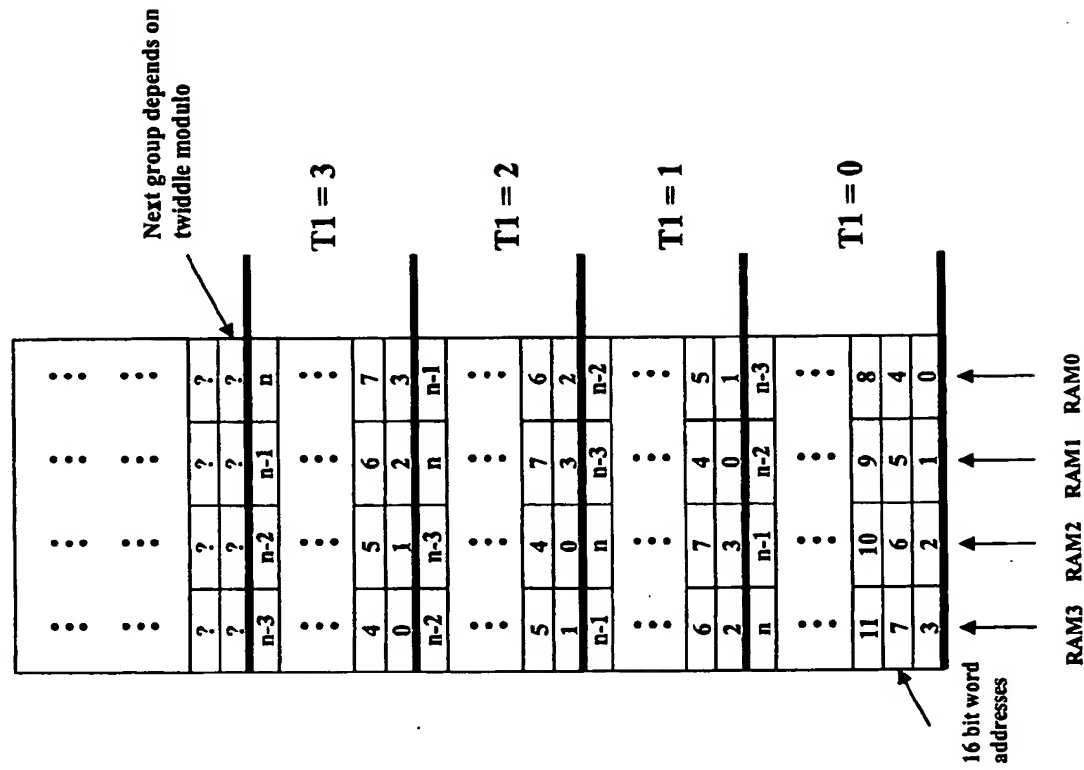
OR



CORRELATOR DATA PATH

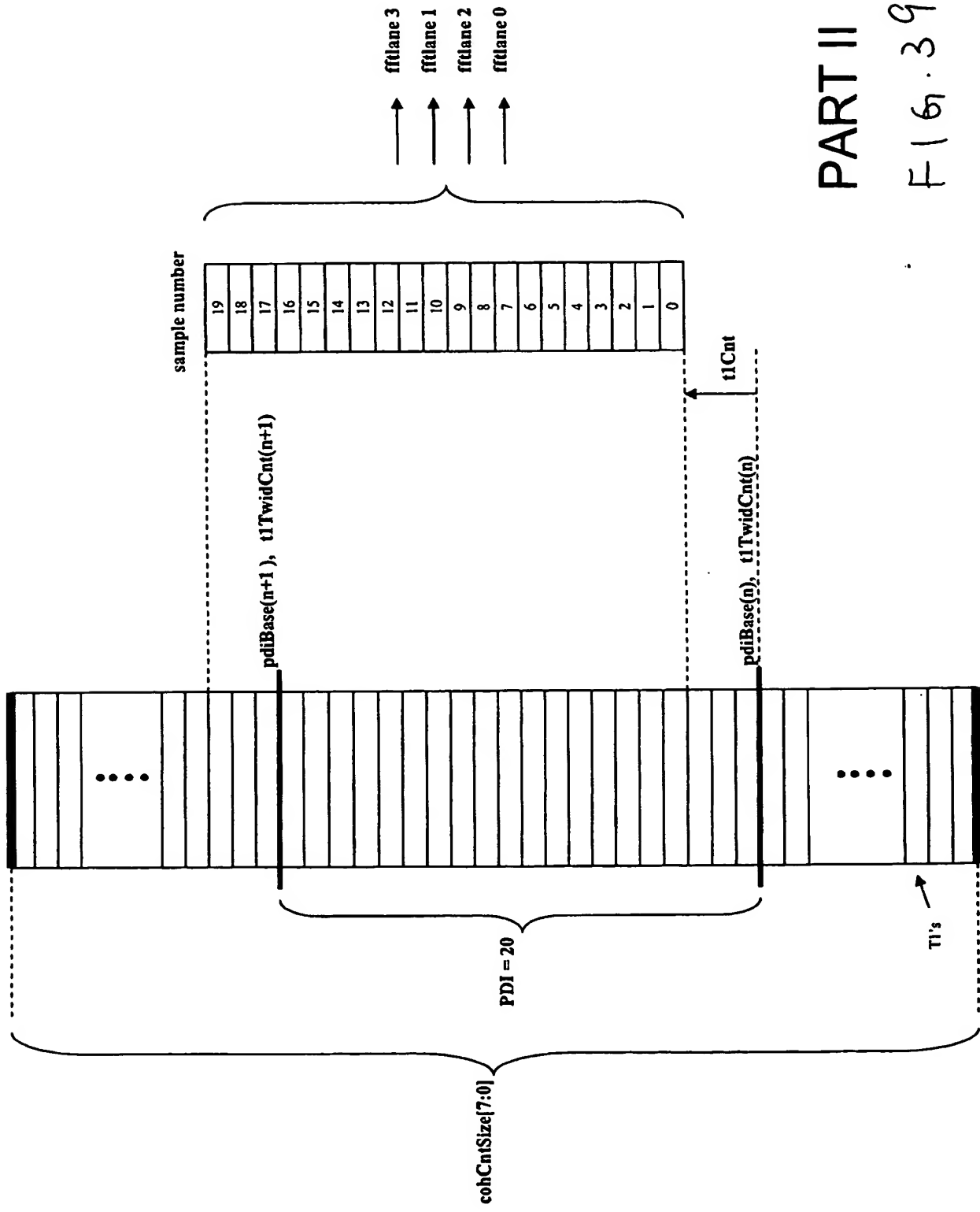
PART II

File. 37



PART II
f16.38

Coherent RAM Data Storage



PART II

F167.39

FFT Access of Coherent RAM (eg. pdi = 20)

fftMode = 0 {4,8}
0,2,1,3 Stride=1, TwiddleModulo = 4

fftMode = 1 {8,8}
fftMode = 2 {8,16}
0,4,2,6 Stride=2, TwiddleModulo = 8
1,5,3,7

fftMode = 3 {16,16}
fftMode = 4 {16,32}
0,4,8,c Stride=4, TwiddleModulo = 16
1,9,5,d
2,a,6,e
3,b,7,f

fftMode = 5 {20,32}
10,12,11,13 Stride=4, TwiddleModulo = 16
- may have collisions
00,08,04,0c
01,09,05,0d
02,0a,06,0e
03,0b,07,0f

FFT - Order Of Data Needed

PART II

Page 40

```

twidModSel = 2, modulo 16 :

twiddled positon = ( t1Twid[3:2] ^ t1Twid[1:0] ^ tapPostion[1:0] )
twiddleSel      = ( t1Twid[3:2] ^ t1Twid[1:0] )

twidModSel = 1, modulo 8 :

twiddled positon = ( (1'b0, t1Twid[2]) ^ t1Twid[1:0] ^ tapPostion[1:0] )
twiddleSel      = ( (1'b0, t1Twid[2]) ^ t1Twid[1:0] )

twidModSel = 0, modulo 4 :

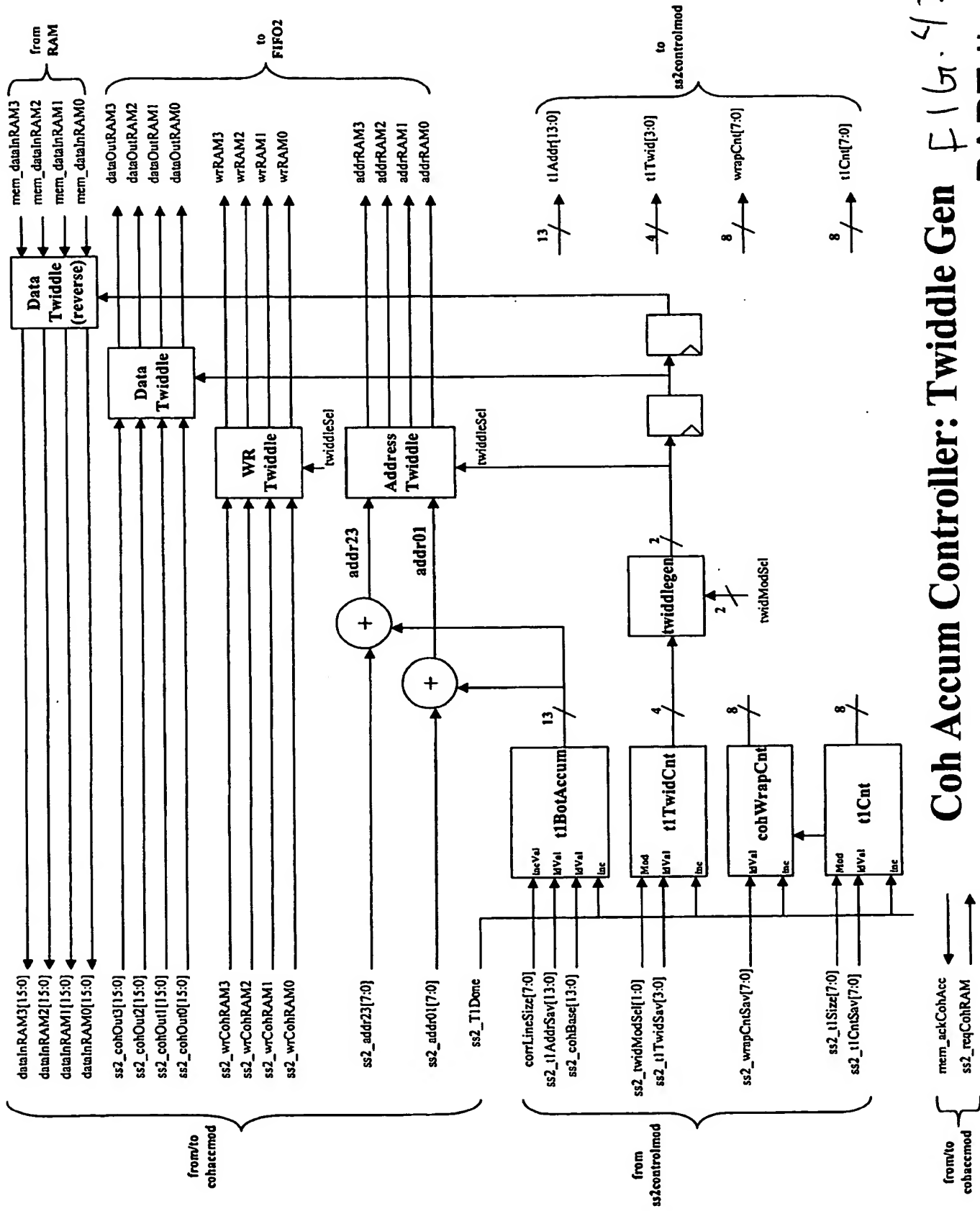
twiddled positon = ( (1'b0, 1'b0) ^ t1Twid[1:0] ^ tapPostion[1:0] )
twiddleSel      = ( (1'b0, 1'b0) ^ t1Twid[1:0] )

```

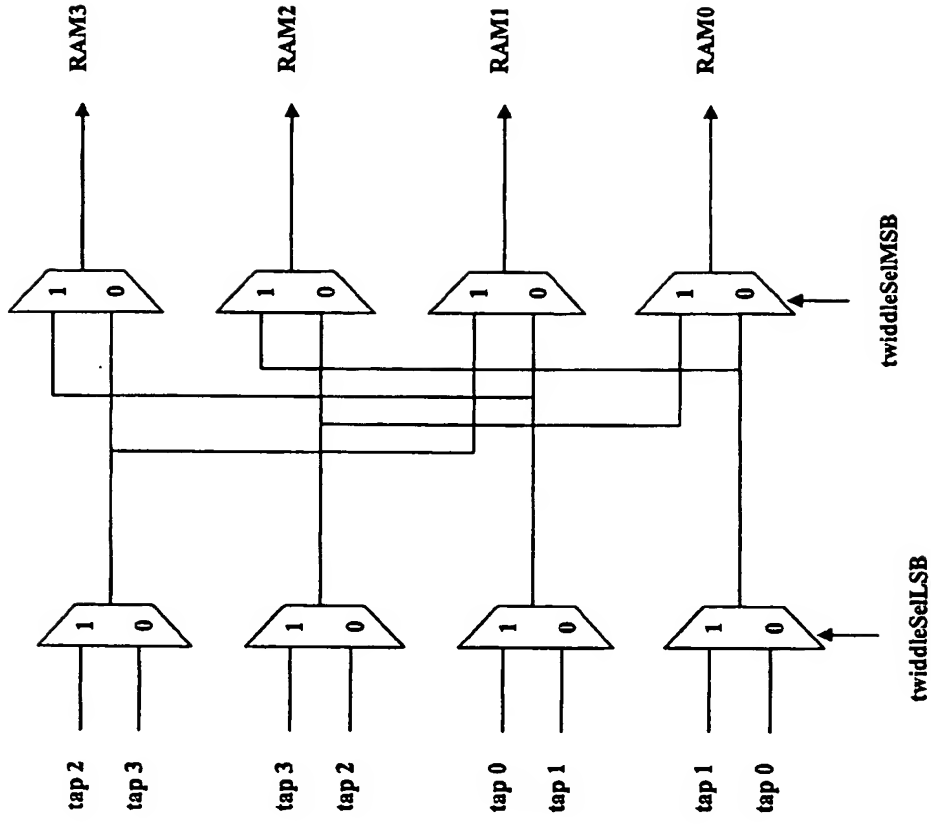
Coherent RAM Input Twiddle Select

PART II

F167.41



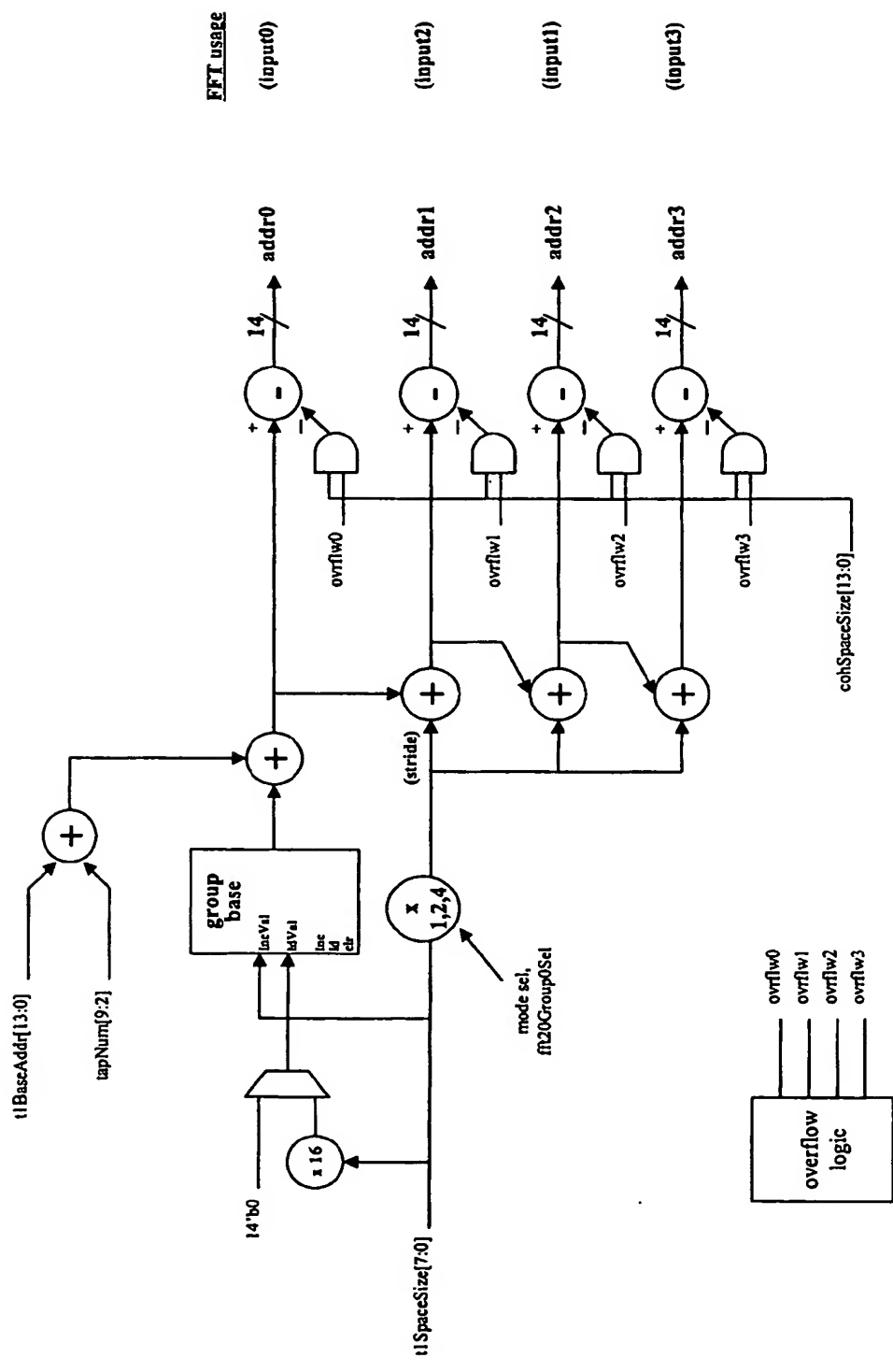
516.42
PART II



Twiddle Mux Implementation

PART II

File 43



FFT usage

(input0)

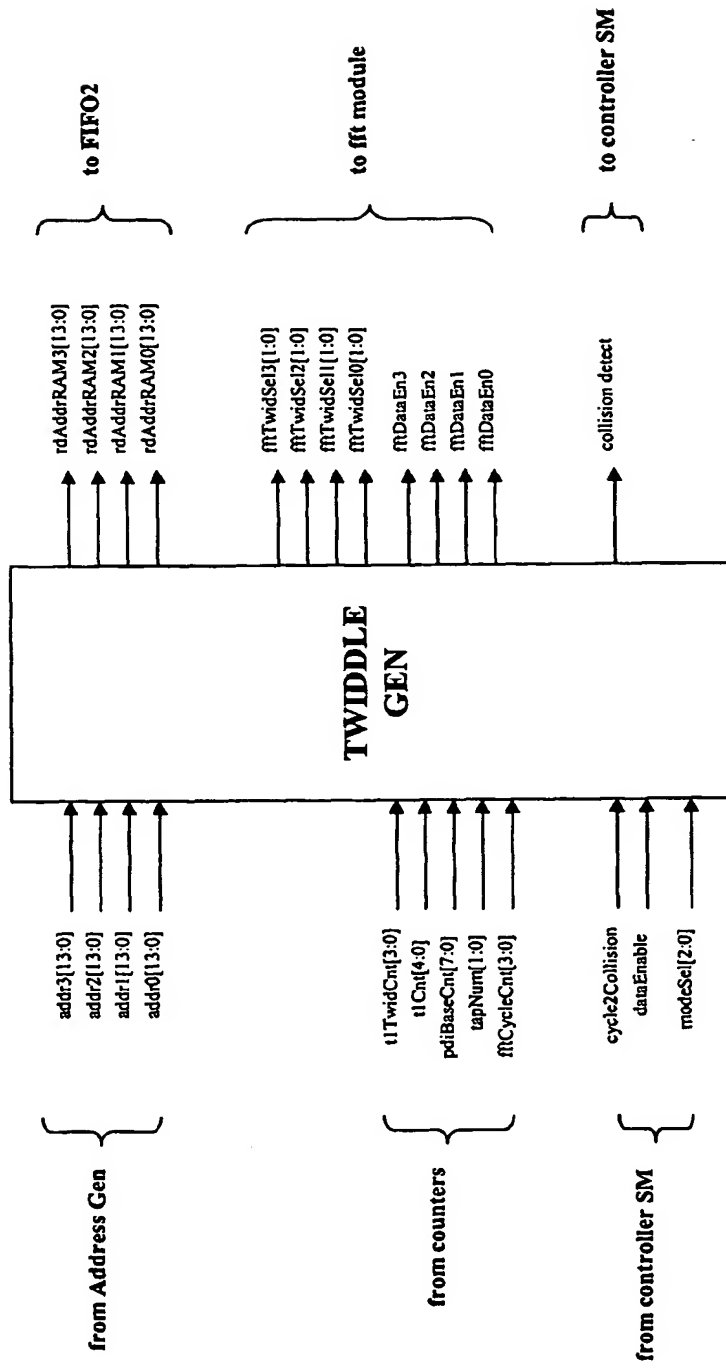
(input2)

(input1)

(input3)

PART II
Fig. 44

FIFO2 Output Address Generation



FFT Address Twiddling

PART II

F-16.45

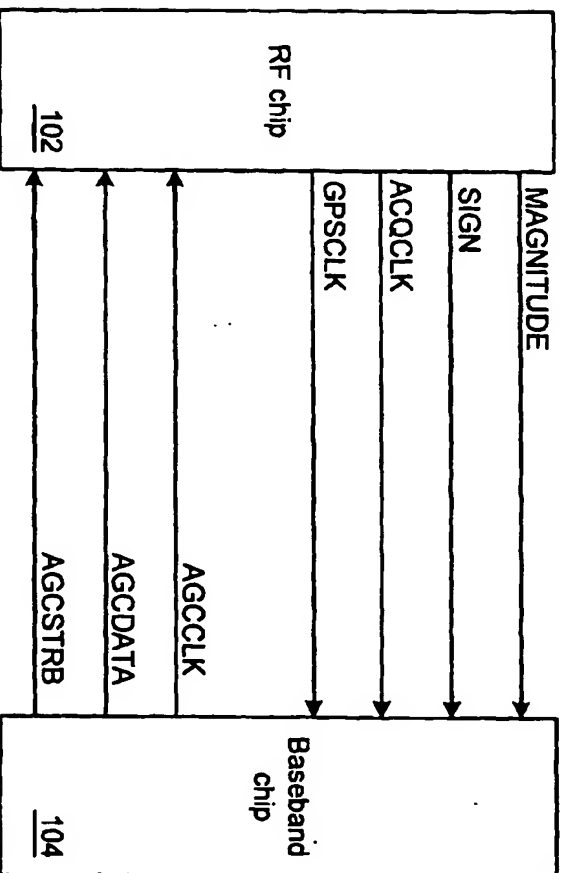


Figure 1 of Appendix A

Prior Art

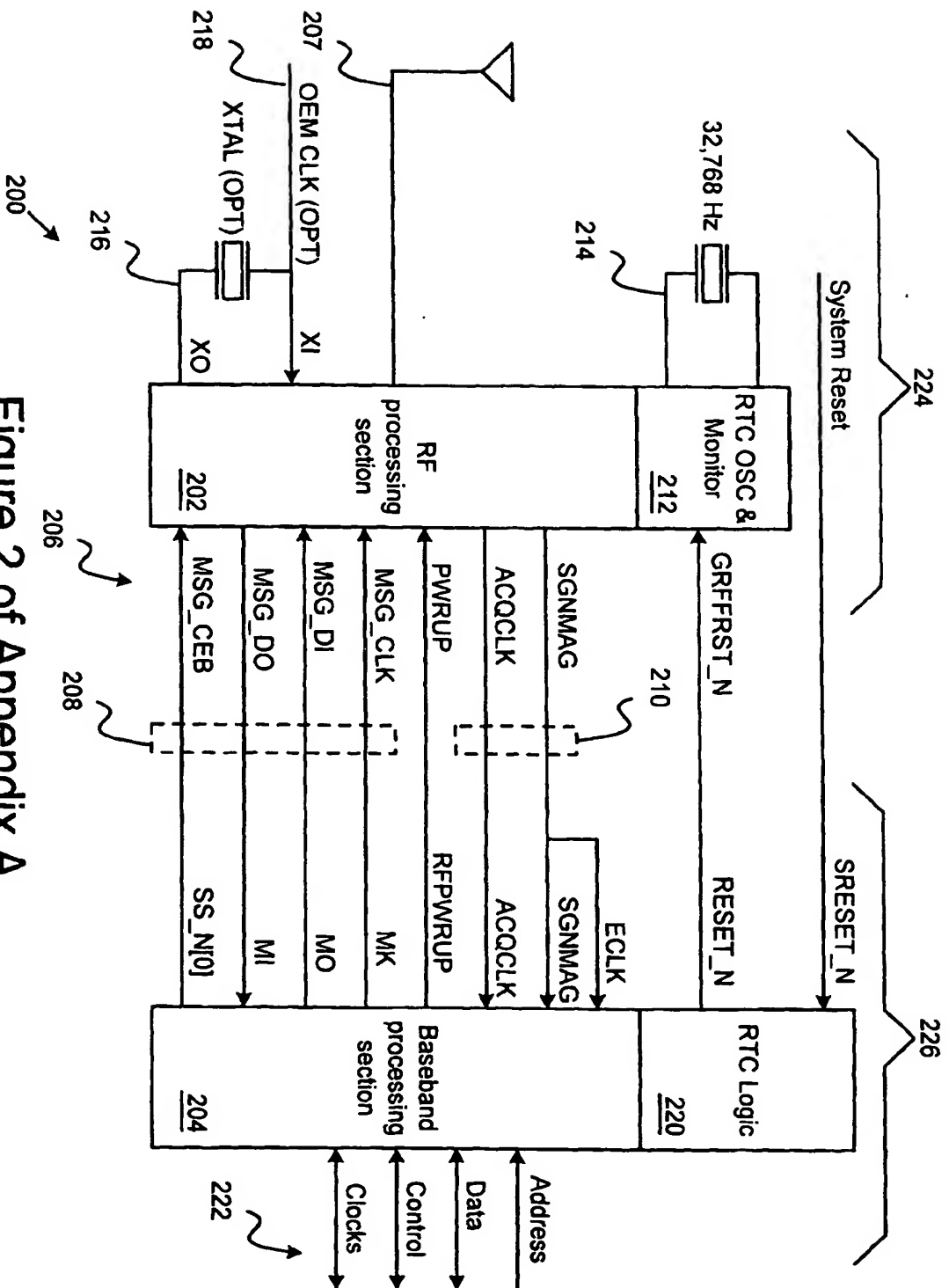


Figure 2 of Appendix A

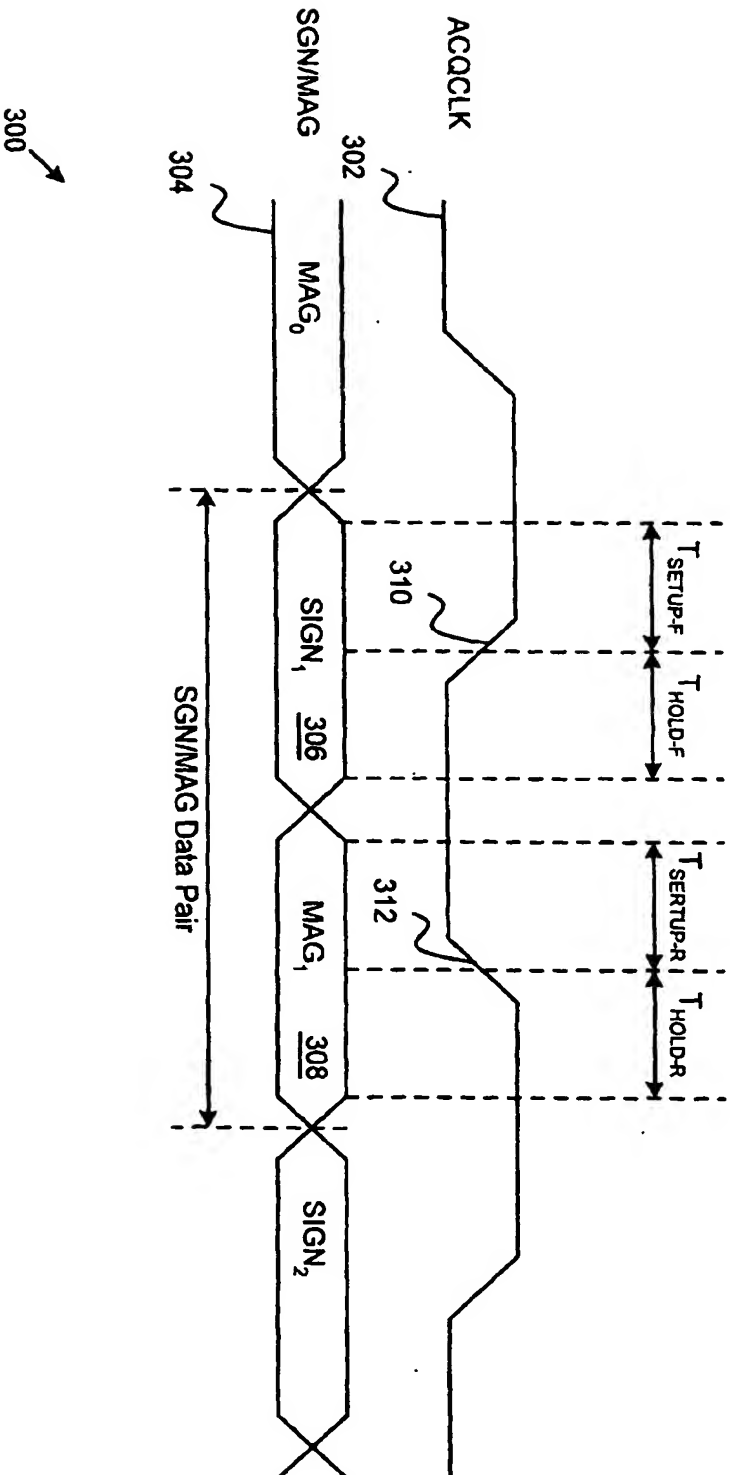


Figure 3 of Appendix A

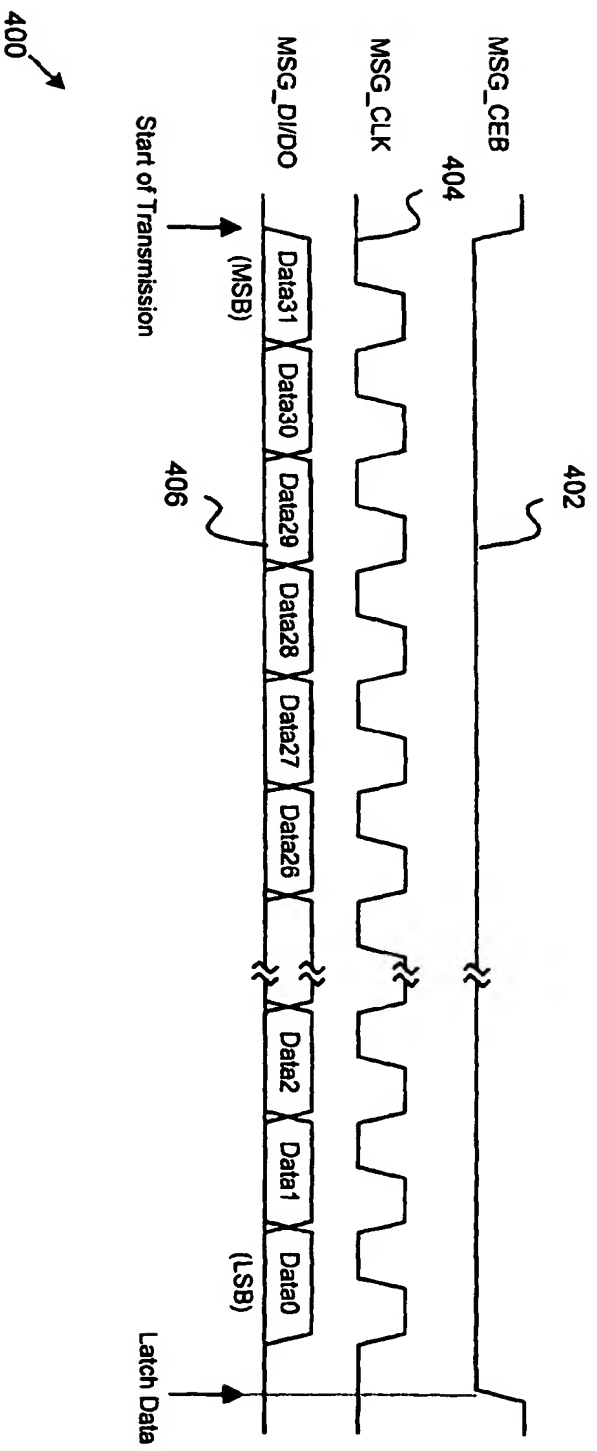


Figure 4 of Appendix A

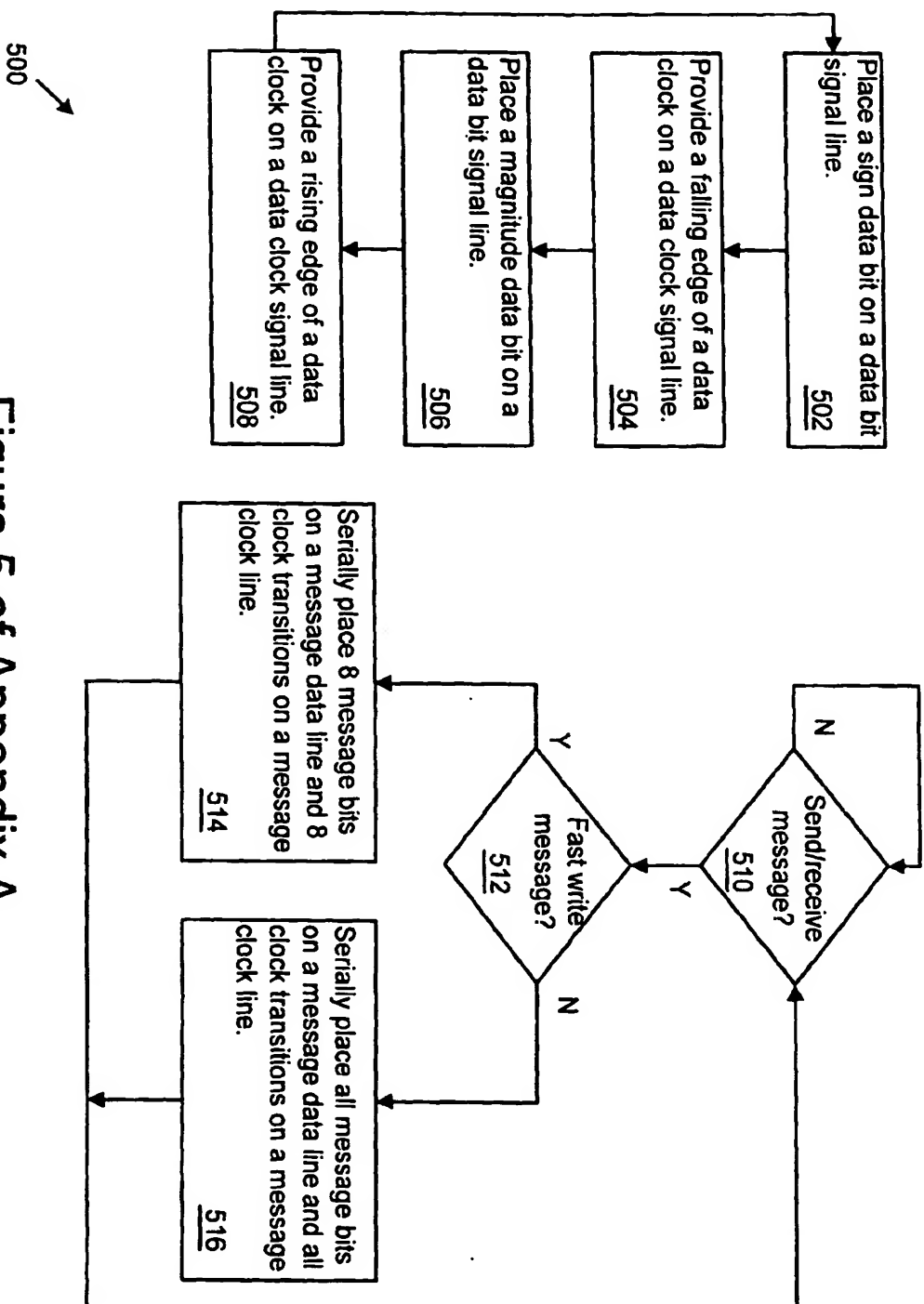


Figure 5 of Appendix A

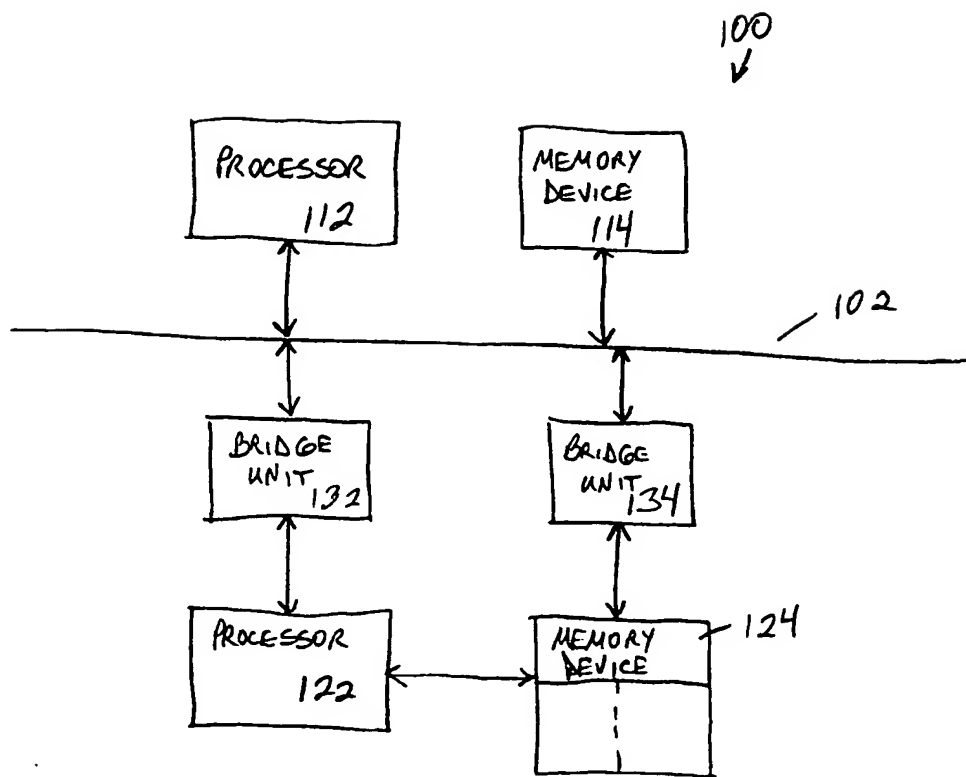
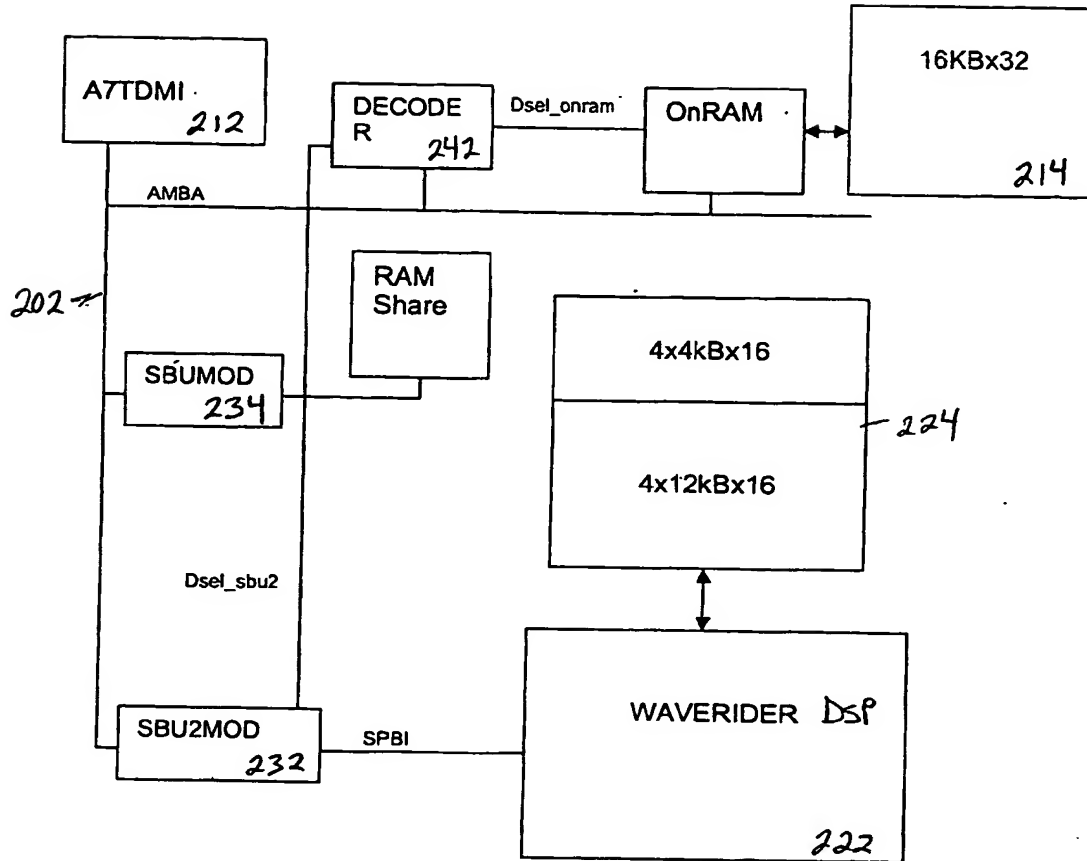


FIGURE 1
APPENDIX B

APPENDIX B

Figure 2

200
↓



APPENDIX B

Figure 3

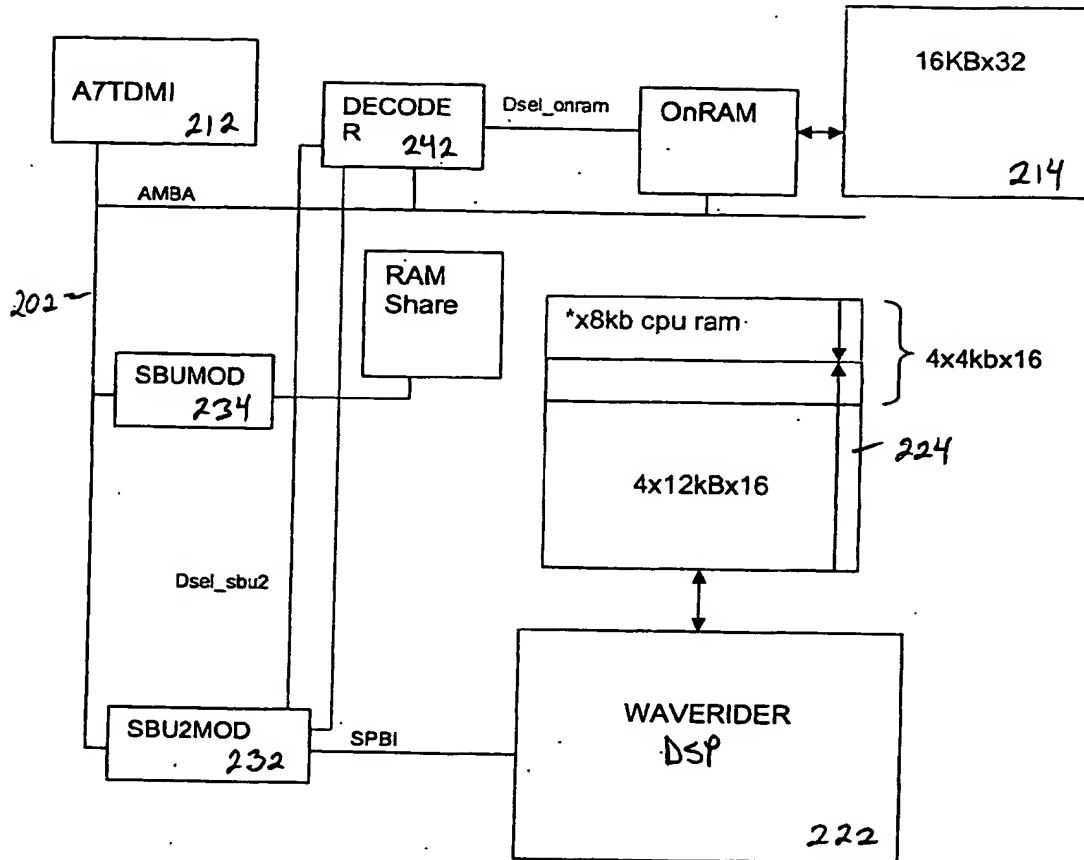
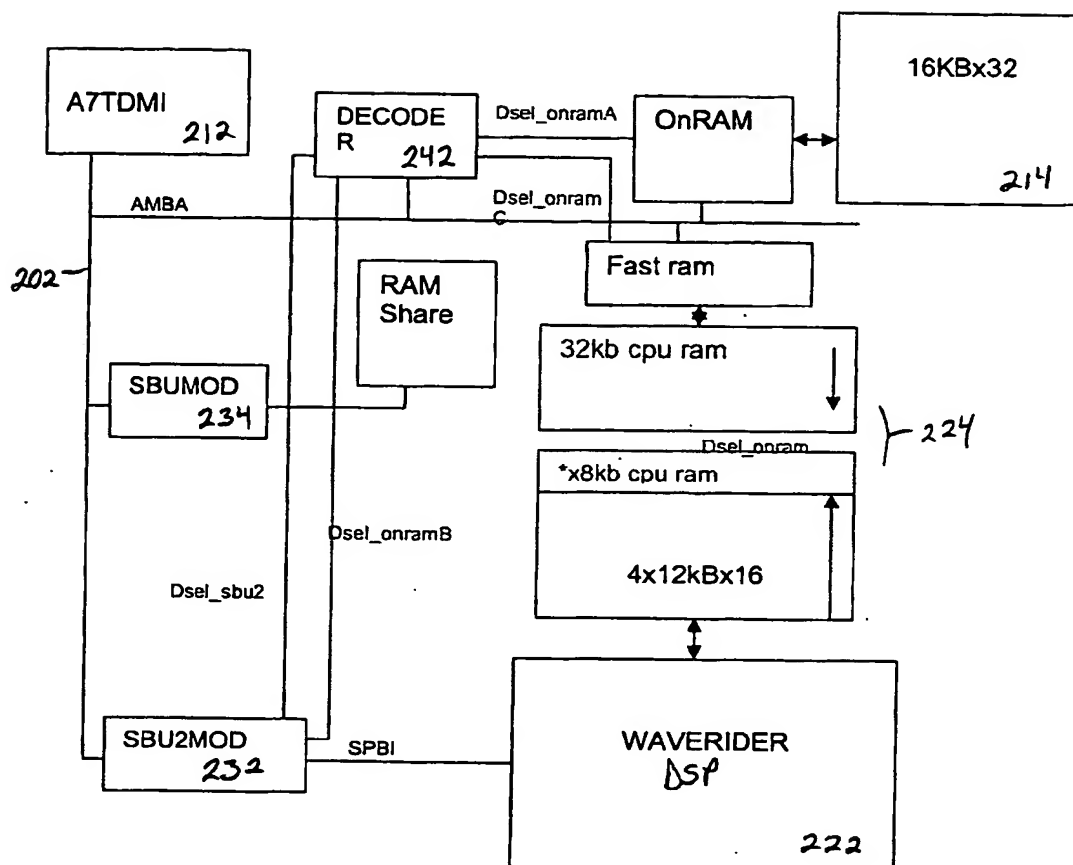
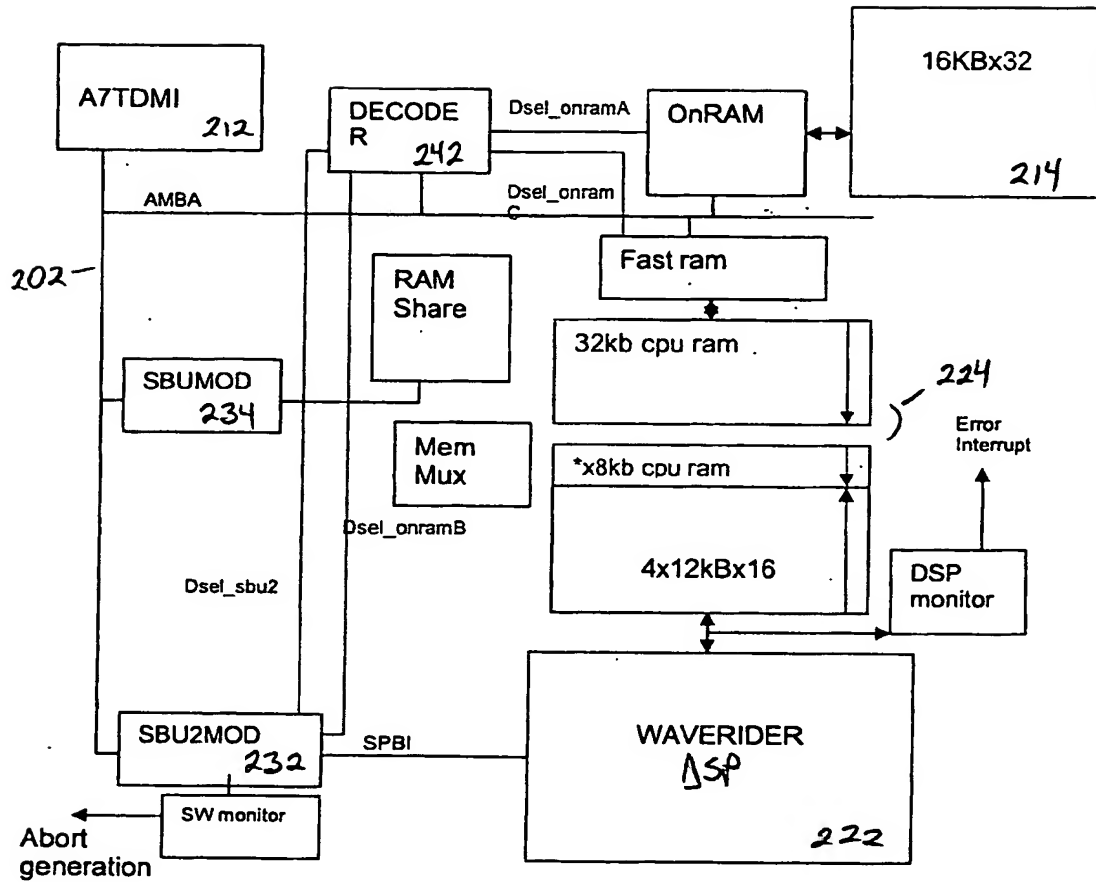


Figure 4



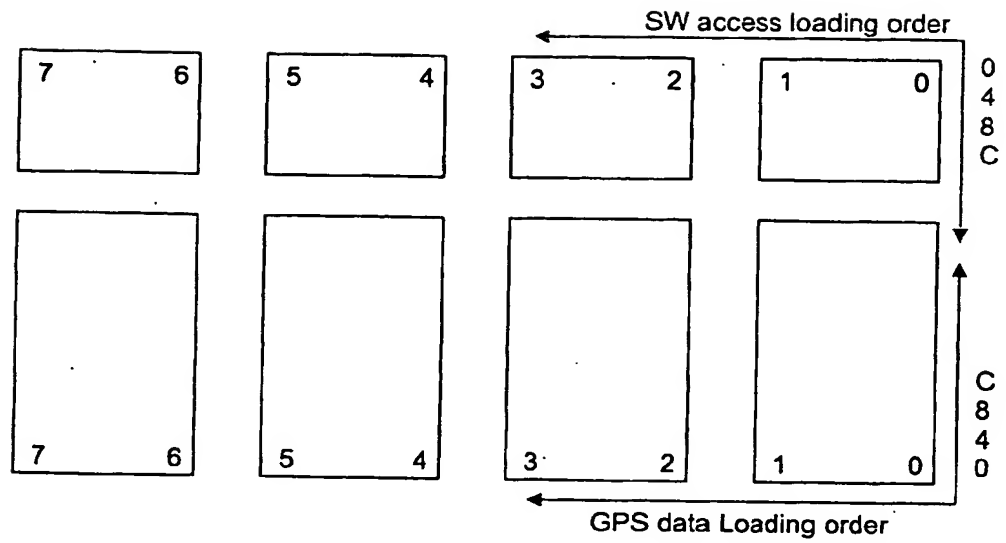
APPENDIX B

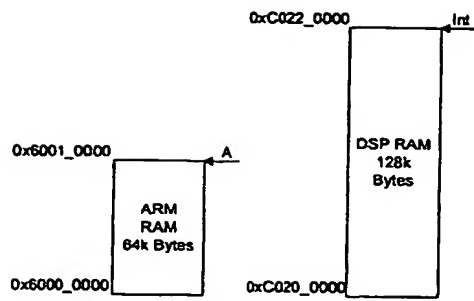
Figure 5



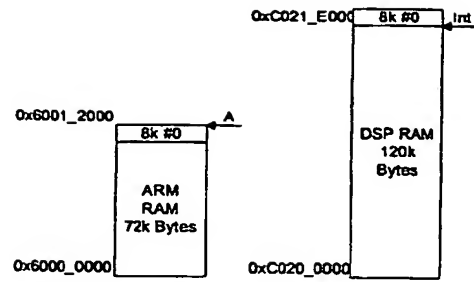
APPENDIX B

Figure 6

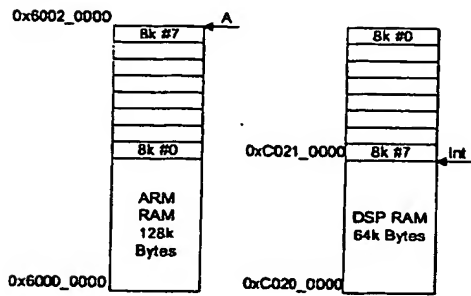




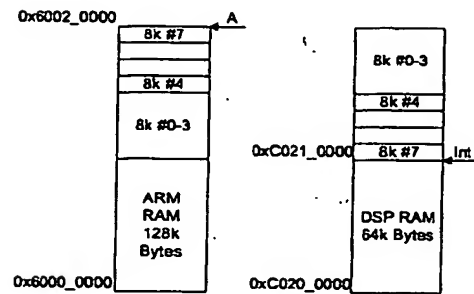
(a) At boot up



(b) First slow 8k segment mapped from DSP to ARM address space through bridge



(c) Zero to eight slow 8k segments mapped from DSP to ARM address space through bridge



(d) Fast access 32k switched from DSP to ARM bus and 0 to 4 slow 8k segments mapped from DSP to ARM address space through bridge

FIGURE 7
APPENDIX B

RAM_CTL: (address = 0xC0000000)

	15	14	13	12	11	10	9	8
	R	R	R	R	R	R	R	RW
Reset value	0	0	0	0	0	0	0	0
	-	-	-	-	-	-	-	-

	7	6	5	4	3	2	1	0
	RW	RW	RW	RW	RW	RW	RW	RW
Reset value	0	0	0	0	0	0	0	0
		SWI_ENB	MAP_BLK[2]	MAP_BLK[1]	MAP_BLK[0]	DSP64K_MAP_ENB	EN_CPU_WAB	EN_CPU_RAB

Figure 8

APPENDIX B

RAM_STA: (address = 0xC0000004)

	15	14	13	12	11	10	9	8
	R	R	R	R	R	R	R	R
Reset value	0	0	0	0	0	0	0	0
	-	-	-	-	-	-	-	-

	7	6	5	4	3	2	1	0
	R	R	R	R	R	R	RW	RW
Reset value	0	0	0	0	0	0	0	0
	-	-	-	-	-	-	CPUW VIO	CPUR VIO

Figure 9

APPENDIX B

DSP_ADDR: (address = 0xC0000008)

	15	14	13	12	11	10	9	8
	R	R	R	R	R	R	R	R
Reset value	0	0	0	0	0	0	0	0
	DSP_ADDR [15]

	7	6	5	4	3	2	1	0
	R	R	R	R	R	R	R	R
Reset value	0	0	0	0	0	0	0	0
	DSP_ADDR [0]

Figure 10

APPENDIX B

APPENDIX B

Figure 11A

Number of 8Kbyte Blocks Mapped	DSP32K_SWI_ENB	DSP64K_MAP_ENB	MAP_BLK[2:0]	DSP Address Range	CPU Normal Mapped SBU2 Address Range	CPU Soft Mapped SBU2 Address Range	CPU Hard Switch ASB Address Range
1	0	0	XXX	0x0000_0000–0x0001_FFFF	0xC020_0000–0xC021_FFFF	NA	NA
	1	X	XXX	0x0000_0000–0x0001_7FFF	0xC021_8000–0xC021_FFFF	NA	0x6001_0000–0x6001_7FFF
	0	1	000	0x0000_0000–0x0001_DFFF	0xC021_E000–0xC021_FFFF	0x6001_0000–0x6001_1FFF	NA
2	0	0	XXX	0x0000_0000–0x0001_FFFF	0xC020_0000–0xC021_FFFF	NA	NA
	1	X	XXX	0x0000_0000–0x0001_7FFF	0xC021_8000–0xC021_FFFF	NA	0x6001_0000–0x6001_7FFF
	0	1	001	0x0000_0000–0x0001_BFFF	0xC021_C000–0xC021_FFFF	0x6001_0000–0x6001_3FFF	NA
3	0	0	XXX	0x0000_0000–0x0001_FFFF	0xC020_0000–0xC021_FFFF	NA	NA
	1	X	XXX	0x0000_0000–0x0001_7FFF	0xC021_8000–0xC021_FFFF	NA	0x6001_0000–0x6001_7FFF
	0	1	010	0x0000_0000–0x0001_9FFF	0xC021_A000–0xC021_FFFF	0x6001_0000–0x6001_5FFF	NA
4	0	0	XXX	0x0001_0000–0x0001_FFFF	0xC020_0000–0xC021_FFFF	NA	NA
	1	X	XXX	0x0000_0000–0x0001_7FFF	0xC021_8000–0xC021_FFFF	NA	0x6001_0000–0x6001_7FFF
	0	1	011	0x0000_0000–0x0001_7FFF	0xC021_8000–0xC021_FFFF	0x6001_0000–0x6001_7FFF	NA
5	0	0	XXX	0x0001_0000–0x0001_FFFF	0xC020_0000–0xC021_FFFF	NA	NA
	1	0	XXX	0x0000_0000–0x0001_7FFF	0xC021_8000–0xC021_FFFF	NA	0x6001_0000–0x6001_7FFF
	1	1	100	0x0000_0000–0x0001_5FFF	0xC021_6000–0xC021_FFFF	0x6001_8000–0x6001_9FFF	0x6001_6000–0x6001_7FFF
	0	1	100	0x0000_0000–0x0001_5FFF	0xC021_6000–0xC021_FFFF	0x6001_0000–0x6001_9FFF	NA
6	0	0	XXX	0x0001_0000–0x0001_FFFF	0xC020_0000–0xC021_FFFF	NA	NA
	1	0	XXX	0x0000_0000–0x0001_7FFF	0xC021_8000–0xC021_FFFF	NA	0x6001_0000–0x6001_7FFF
	1	1	101	0x0000_0000–0x0001_3FFF	0xC021_4000–0xC021_FFFF	0x6001_8000–0x6001_BFFF	0x6001_0000–0x6001_7FFF
	0	1	101	0x0000_0000–0x0001_3FFF	0xC021_4000–0xC021_FFFF	0x6001_0000–0x6001_BFFF	NA
7	0	0	XXX	0x0001_0000–0x0001_FFFF	0xC020_0000–0xC021_FFFF	NA	NA

Number of 8Kbyte Blocks Mapped	DSP32K_SWL_ENB	DSP64K_MAP_ENB	MAP_BLK[2:0]	DSP Address Range	CPU Normal Mapped SBU2 Address Range	CPU Soft Mapped SBU2 Address Range	CPU Hard Switch ASB Address Range
	1	0	XXX	0x0000_0000–0x0001_7FFF	0xC021_8000–0xC021_FFFF	NA	0x6001_0000–0x6001_7FFF
	1	1	110	0x0000_0000–0x0001_1FFF	0xC021_2000–0xC021_FFFF	0x6001_8000–0x6001_DFFF	0x6001_0000–0x6001_7FFF
	0	1	110	0x0000_0000–0x0001_1FFF	0xC021_2000–0xC021_FFFF	0x6001_0000–0x6001_DFFF	NA
8	0	0	XXX	0x0001_0000–0x0001_FFFF	0xC020_0000–0xC021_FFFF	NA	NA
	1	0	XXX	0x0000_0000–0x0001_7FFF	0xC021_8000–0xC021_FFFF	NA	0x6001_0000–0x6001_7FFF
	1	1	111	0x0000_0000–0x0000_FFFF	0xC021_0000–0xC021_FFFF	0x6001_8000–0x6001_EFFF	0x6001_0000–0x6001_7FFF
	0	1	111	0x0000_0000–0x0000_FFFF	0xC021_0000–0xC021_FFFF	0x6001_0000–0x6001_FFFF	NA

Figure 11B

APPENDIX B

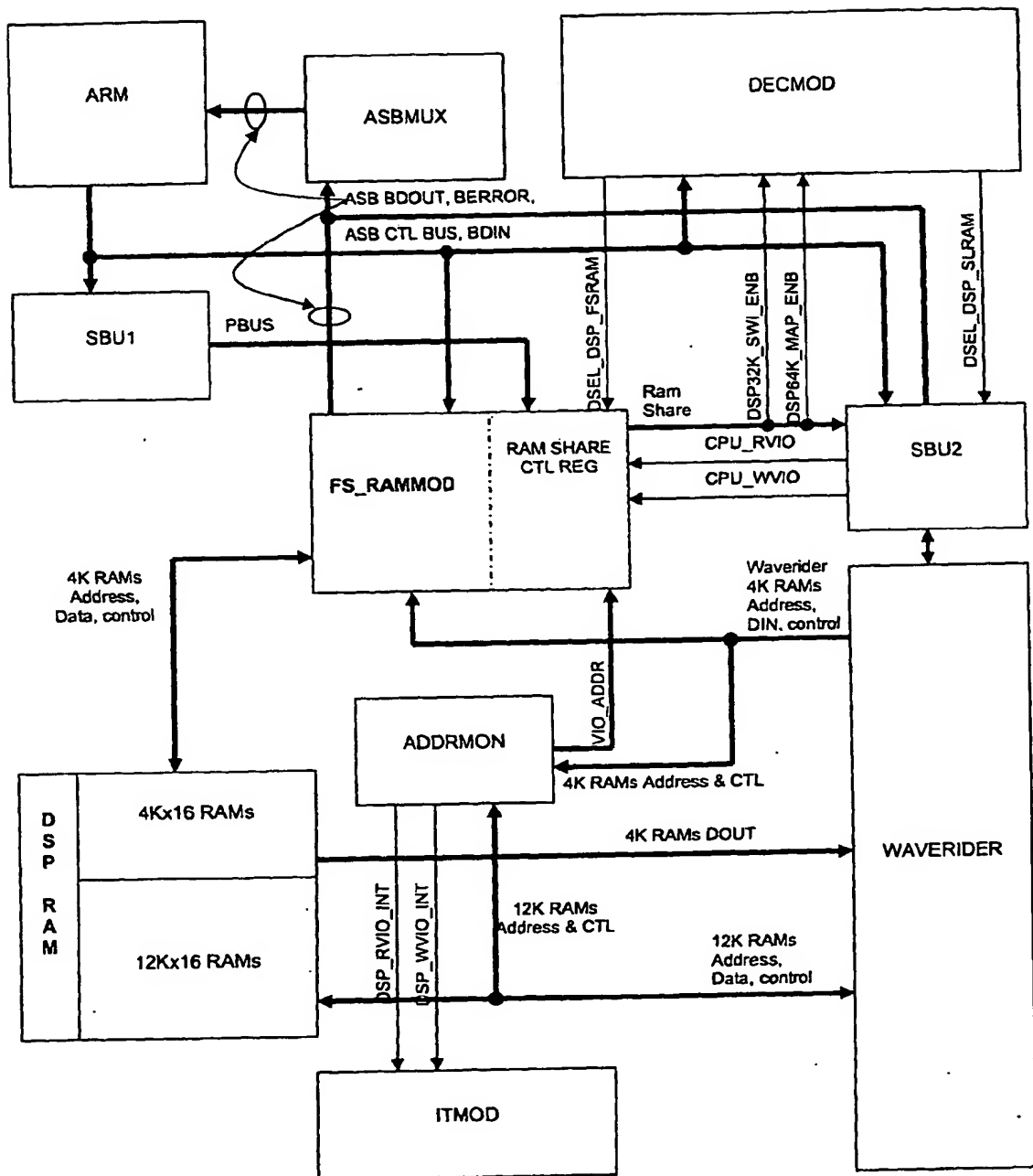


FIGURE 12
APPENDIX B

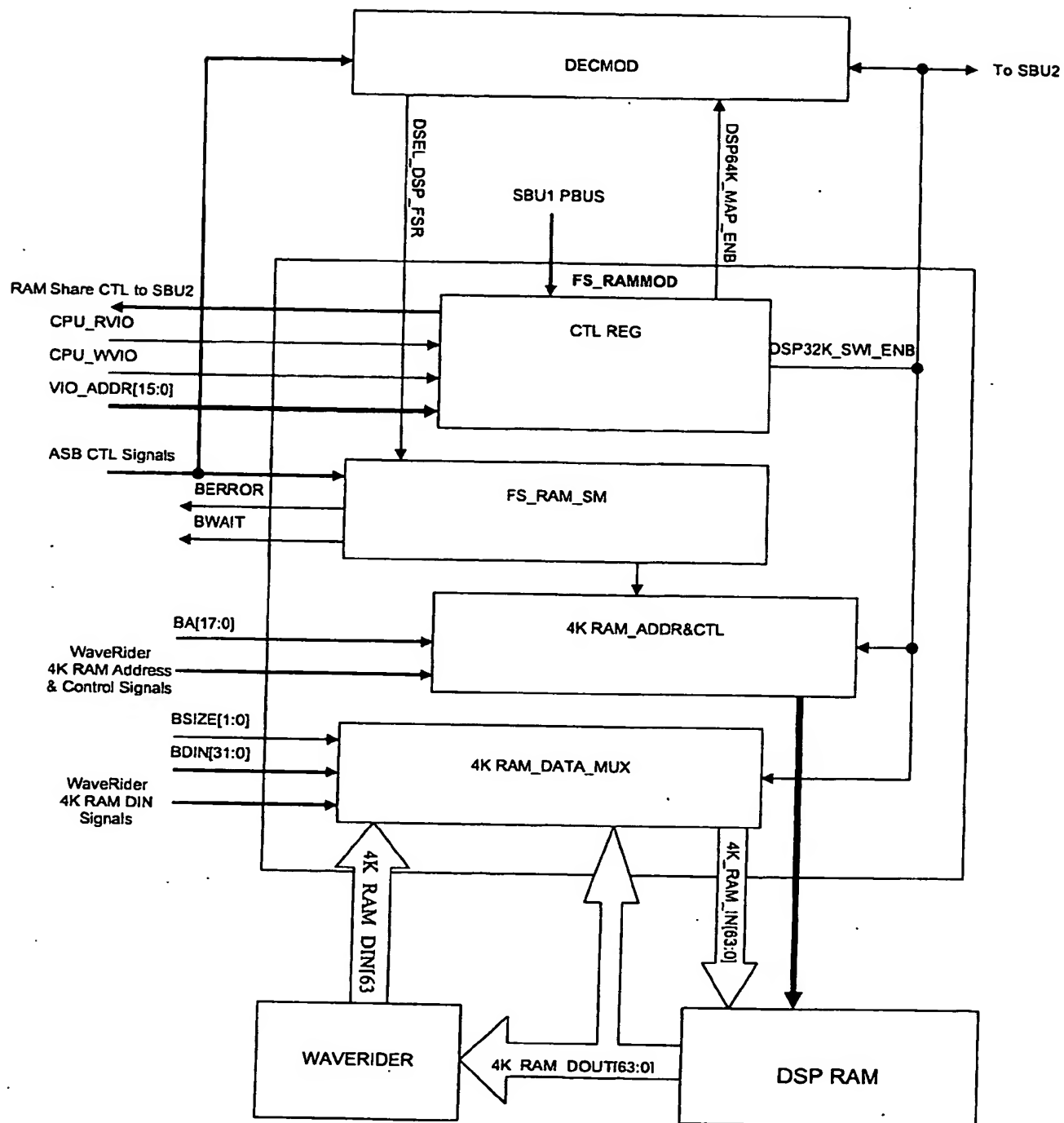


FIGURE 13
APPENDIX B

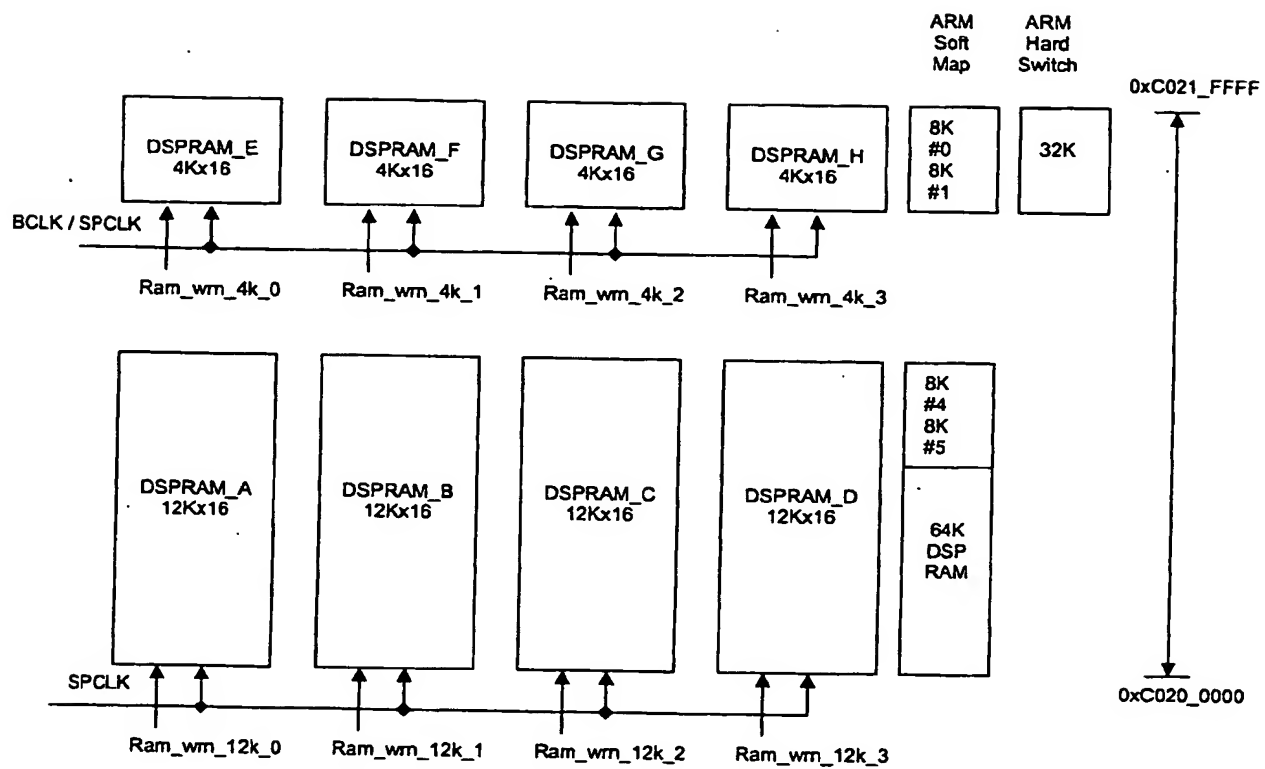


FIGURE 14
APPENDIX B

APPENDIX B

Figur 15A

VES RAMMOD Module I/O Table				
Name	Bits	I/O	Description	SRG/TEST
ATPGMODE	1	Input	ATPG scan chain mode	TESTMOD
BA[16:0]	17	Input	ASB address bus	ASB
BCLK	1	Input	CPU bus clock	ASB
BDIN[31:0]	32	Input	ASB data input bus	ASB
BDOUT[31:0]	32	Output	ASB data output bus	ASBMUX
BERROR	1	Output	ASB error signal (always logic '0')	ASBMUX
BRES_N	1	Input	Master reset, active low	ASB
BSIZE[1:0]	2	Input	ASB Byte selects	ASB
BWAIT	1	Output	ASB wait signal	ASBMUX
BWRITE	1	Input	ASB read/write	ASB
DSEL_FSRAM	1	Input	ASB DSEL for fast access DSP RAM	DECMOD
PA[1:0]	2	Input	Peripheral address bus	SBU1
PDIN	16	Input	Peripheral write data bus	SBU1
PSEL_RAMCTL	1	Input	Peripheral select for DSP RAM control registers	SBU1
PSTB	1	Input	Peripheral bus read strobe	SBU1
PWRITE	1	Input	Peripheral bus write access	SBU1
CPU RVIO	1	Input	CPU RAM read violation status	SBU2
CPU WVIO	1	Input	CPU RAM write violation status	SBU2
EN_CPU_RAB	1	Output	Enable abort on CPU RAM read violation	SBU2
EN_CPU_WAB	1	Output	Enable abort on CPU RAM write violation	SBU2
MAP_BLK[2:0]	1	Output	Soft mapped DSP RAM boundary encoded bits for CPU	SBU2
DSP64K_MAP_ENB	1	Output	Enable CPU soft mapping of upper 64K of DSP RAM through SBU2 bridge	SBU2 DECMOD
DSP32K_SWI_ENB	1	Output	Enable hard switch of DSP upper 32Kbyte RAM for CPU direct access.	SBU2 DECMOD
DSPVIO_ADDR[15:0]	16	Input	DSP read/write violation address	ADDRMON
WR_4K_CSN	1	Input	DSP 4K RAMs' chip select	WAVERIDER
WR_4K_WRN0	1	Input	DSP 4K RAM E write enable	WAVERIDER
WR_4K_WRN1	1	Input	DSP 4K RAM F write enable	WAVERIDER
WR_4K_WRN2	1	Input	DSP 4K RAM G write enable	WAVERIDER
WR_4K_WRN3	1	Input	DSP 4K RAM H write enable	WAVERIDER
WR_4K_ADD0[11:0]	12	Input	DSP 4K RAM E address	WAVERIDER
WR_4K_ADD1[11:0]	12	Input	DSP 4K RAM F address	WAVERIDER
WR_4K_ADD2[11:0]	12	Input	DSP 4K RAM G address	WAVERIDER
WR_4K_ADD3[11:0]	12	Input	DSP 4K RAM H address	WAVERIDER
WR_4K_DIN0[15:0]	16	Input	DSP 4K RAM E input data	WAVERIDER
WR_4K_DIN1[15:0]	16	Input	DSP 4K RAM F input data	WAVERIDER
WR_4K_DIN2[15:0]	16	Input	DSP 4K RAM G input data	WAVERIDER
WR_4K_DIN3[15:0]	16	Input	DSP 4K RAM H input data	WAVERIDER
RAM4K_CSN	1	Output	DSP 4K RAM E,F,G,H chip select	DSP RAM
RAM4K_WRN0	1	Output	DSP 4K RAM E write enable	DSP RAM E
RAM4K_WRN1	1	Output	DSP 4K RAM F write enable	DSP RAM F
RAM4K_WRN2	1	Output	DSP 4K RAM G write enable	DSP RAM G

DSP RAM I/O Bit I/O Table				
Name	Bit	I/O	Description	SRC/DEST
RAM4K WRN3	1	Output	DSP 4K RAM H write enable	DSP RAM H
RAM4K ADD0[11:0]	12	Output	DSP 4K RAM E address	DSP RAM E
RAM4K ADD1[11:0]	12	Output	DSP 4K RAM F address	DSP RAM F
RAM4K ADD2[11:0]	12	Output	DSP 4K RAM G address	DSP RAM G
RAM4K ADD3[11:0]	12	Output	DSP 4K RAM H address	DSP RAM H
RAM4K DIN0[15:0]	16	Output	DSP 4K RAM E input data	DSP RAM E
RAM4K DIN1[15:0]	16	Output	DSP 4K RAM F input data	DSP RAM F
RAM4K DIN2[15:0]	16	Output	DSP 4K RAM G input data	DSP RAM G
RAM4K DIN3[15:0]	16	Output	DSP 4K RAM H input data	DSP RAM H
RAM4K DOUT0[15:0]	16	Input	DSP 4K RAM E output data	DSP RAM E
RAM4K DOUT1[15:0]	16	Input	DSP 4K RAM F output data	DSP RAM F
RAM4K DOUT2[15:0]	16	Input	DSP 4K RAM G output data	DSP RAM G
RAM4K DOUT3[15:0]	16	Input	DSP 4K RAM H output data	DSP RAM H

Figure 15B

APPENDIX B

Document made available under the Patent Cooperation Treaty (PCT)

International application number: PCT/US04/028926

International filing date: 02 September 2004 (02.09.2004)

Document type: Certified copy of priority document

Document details: Country/Office: US
Number: 60/499,961
Filing date: 02 September 2003 (02.09.2003)

Date of receipt at the International Bureau: 07 April 2005 (07.04.2005)

Remark: Priority document submitted or transmitted to the International Bureau in compliance with Rule 17.1(a) or (b)



World Intellectual Property Organization (WIPO) - Geneva, Switzerland
Organisation Mondiale de la Propriété Intellectuelle (OMPI) - Genève, Suisse

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☒ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☒ **FADED TEXT OR DRAWING**
- ☐ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☐ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☐ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER:** _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.